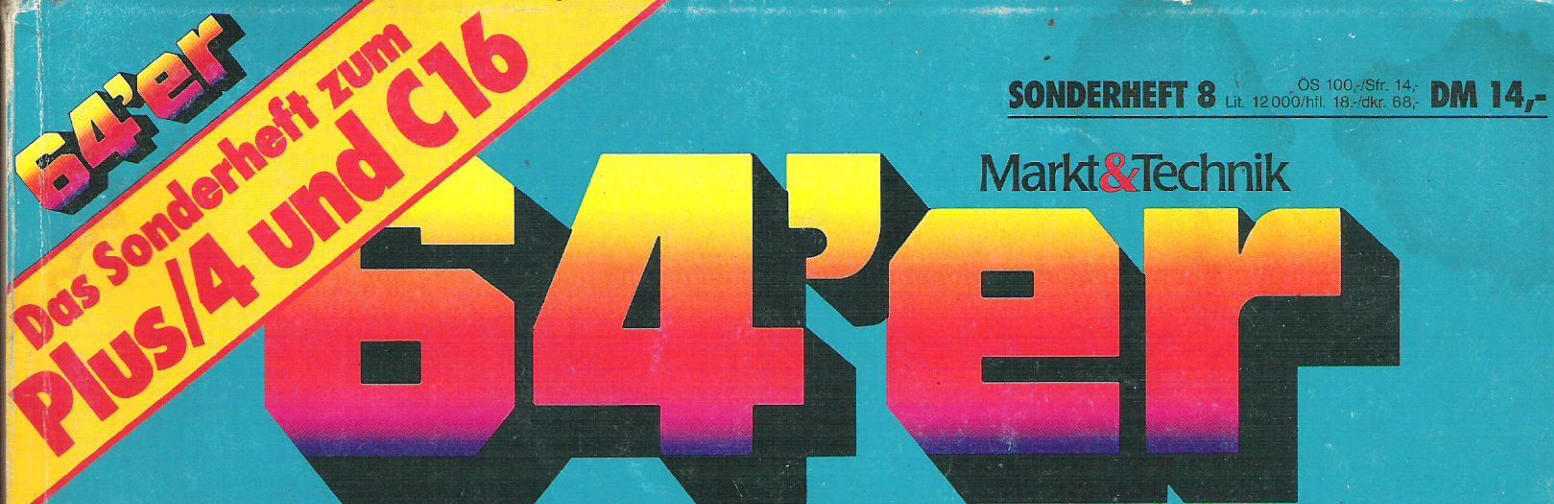


SONDERHEFT 8 ÖS 100,-/Sfr. 14,- Lit. 12 000/hfl. 18,-/dkr. 68,- DM 14,-

C 16
C 116
Plus/4
VC 20

Wichtige Grundlagen

★ Basic
★ Grafik
★ Maschinensprache

Jede Menge Programme zum Abtippen

★ Action-Spiele
★ fantastische Grafik
★ Programmierhilfen
★ Kopierprogramme
★ viele Tips und Tricks

Alle Programme auch auf Kassette und Diskette erhältlich

Alles für C 16 und Plus/4



Vor kurzem kam ein Redakteur aufgeregt in mein Zimmer hereingestürmt. »Hast Du das schon gesehen?« Er hielt mir die Anzeige einer Kaufhauskette vor die Nase. Ich rieb mir die Augen, als ich genauer hinschaute: Plus/4 mit Floppy 1551 für sagenhafte 499 Mark! »Das ist ja ein Ding« entfuhr es mir. Ein Anruf bei Commodore brachte Klarheit. Kein Druckfehler, keine Zeitungsente, es stimmte. Eine große Anzahl Plus/4 werden zu diesem »kann-man-gar-nicht-glauben«-Preis angeboten und sicherlich im Handumdrehen verkauft sein. Sehr schnell wurden mir die Konsequenzen, die sich daraus ergeben würden, klar. Zusammen mit den C 16/C 116-Besitzern werden daraus leicht mehr als 300000 Computerfreaks, die zum größten Teil keine oder nur wenig Erfahrung mit diesen oder ähnlichen Geräten haben. Weiterhin bedeutete das, daß wir in der 64'er-Redaktion unseren Gerätepark erweitern mußten. Die wichtigste Erkenntnis aber war die, daß Sie, die Besitzer dieser beiden Computer, etwas von uns erwarten werden, nämlich Informationen. Genauer gesagt, Informationen zum Lernen und Listings zum Abtippen. Genau das finden Sie in diesem Sonderheft.

Die Beschäftigung mit Heimcomputern ist in der Regel eine reine Freizeitbeschäftigung. Sie macht Spaß, weil man gefordert wird, jeder nach seinem Wissen und seinen Fähigkeiten, vom Anfänger bis zum Profi. Die Handbücher zum Computer sind mehr oder weniger gute Hilfen zum Einstieg. Schon bald möchte man jedoch mehr wissen. Viele Informationen fehlen nämlich oder sind zu knapp beschrieben. Aus diesem Grund haben wir in diesem Sonderheft einen riesigen Basic-Kurs zum Mitmachen. Die wichtigsten Befehle des C 16 und Plus/4 werden ausführlich und mit vielen Beispielen erklärt. Mit diesem Wissen werden Sie problemlos Basic-Programme schreiben lernen.

Bekanntlich besitzt der C 16/Plus/4 einen eingebauten Maschinensprachemonitor, der vielen Einsteigern wahrscheinlich größere Kopfschmerzen bereiten wird. Wir erklären Ihnen klipp und klar, welche fantastischen Möglichkeiten Sie mit diesem Programm haben. Natürlich auch hier mit vielen Beispielen und genauen Erklärungen. Wer danach noch weiter in die Maschinensprache einsteigen möchte, findet einen Artikel mit sehr wichtigen Informationen zur Ein- und Ausgabe von Zeichen, Wörtern und Zahlen. Selbstverständlich kann man nie alles erklären, Fragen bleiben immer. In diesem Zusammenhang möchte ich Sie noch auf drei Sonderhefte hinweisen, die dieses Sonderheft ausgezeichnet ergänzen:

- das C 16/Plus/4-Sonderheft 3/86: viele Kurse speziell für Einsteiger, hauptsächlich Grafik, Musik, Dateiverwaltung und Maschinensprache. Dazu eine genaue Erklärung der »Innereien« des C 16, dessen Aufbau und Programmierung. Wie in jedem Sonderheft natürlich mit einem großen Listingteil zum Abtippen.
- Grundwissen-Sonderheft 5/86: zwar nicht speziell für den C 16/Plus/4, aber mit vielen Beiträgen, die für alle Commodore-Computer gelten, zum Beispiel »Dateiverwaltung für Einsteiger«, ein Kurs über die Arbeit mit der Floppy und mit Druckern, eine Hilfe bei der Frage »Monitor oder Fernseher?«, etc.
- »Maschinensprache für Anfänger und Fortgeschrittene«, Sonderheft 8/85, ein Standardwerk, das sehr viel Lob geerntet hat und zu dem wir immer noch viele begeisterte Zuschriften erhalten. Sie finden in diesem Heft einen kompletten Maschinensprache-Lernkurs, alle Programme, die man braucht, um in Assembler zu programmieren, jede Menge Tips & Tricks und immer wieder benötigte wichtige Tabellen zum Nachschlagen.

Mit diesen Sonderheften erhalten Sie eine Bibliothek von ausgezeichneten Beiträgen, auf die Sie immer wieder zurückgreifen werden, auch dann, wenn Sie bereits ausgiebige Erfahrungen gesammelt haben. Ich hoffe, Ihnen gefällt dieses Heft. Falls das nicht der Fall sein sollte, ist uns Ihre Kritik ebenso wertvoll. Denn wir möchten, daß Sie rundum zufrieden sind.

(Georg Klinge)

PROGRAMM-SERVICE



Bestellungen in der Schweiz: Markt&Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, Tel. 042/41 56 56
Bestellungen in Österreich: Bücherzentrum Meidling, Schönbrunner Straße 261, A-1120 Wien, Tel. 0222/83 31 96,
Microcomput-ique E. Schiller, Fasangasse 21, A-1030 Wien, Tel. 0222/78 56 61,
Ueberreuter Media Handels- und Verlagsgesellschaft mbH, Alser Straße 24, A-1091 Wien, Tel. 0222/48 15 38-0
Bestellungen aus anderen Ländern bitte per Auslandspostanweisung!

Das Angebot dieser Ausgabe:

Wer keine Zeit oder Lust hat, alle Programme selbst in mühevoller Kleinarbeit abzuschreiben, kann wieder auf den bewährten Programm-Service zurückgreifen. Alle Programme, die mit dem Diskettensymbol im Inhaltsverzeichnis gekennzeichnet sind, gibt's auf Diskette bzw. Kassette.

1 Diskette für VC 20 und C 116
Bestell-Nr. L6 86 S8 CD (sFr. 24,90/öS 299,-*) **DM 29,90***

4 Kassetten für C16/C116
Bestell-Nr. L6 86 S8 KC (sFr. 29,50/öS 349,-*) **DM 34,90***

1 Kassette für VC 20
Bestell-Nr. L6 86 S8 KV (sFr. 17,-/öS 199,-*) **DM 19,90***

* inkl. MwSt. Unverbindliche Preisempfehlung

Programme aus früheren Ausgaben:

64'er-Ausgabe 7/86
Bestell-Nr. L6 86 07D Diskette
DM 29,90* (sFr. 24,90/öS 299,-*)

64'er-Ausgabe 6/86
Bestell-Nr. L6 86 06D Diskette
DM 29,90* (sFr. 24,90/öS 299,-*)

64'er-Ausgabe 5/86
Bestell-Nr. L6 86 05D Diskette
DM 29,90* (sFr. 24,90/öS 299,-*)

64'er-Ausgabe 4/86
Bestell-Nr. L6 86 04D Diskette
DM 29,90* (sFr. 24,90/öS 299,-*)

64'er-Ausgabe 3/86
Bestell-Nr. L6 86 03D Diskette
DM 29,90* (sFr. 24,90/öS 299,-*)

64'er-Ausgabe 2/86
Bestell-Nr. L6 86 02D Diskette
DM 29,90* (sFr. 24,90/öS 299,-*)

64'er-Ausgabe 1/86
Bestell-Nr. L6 86 01D Diskette
DM 29,90* (sFr. 24,90/öS 299,-*)

64'er-Ausgabe 12/85
Bestell-Nr. L6 85 12D Diskette
DM 29,90* (sFr. 24,90/öS 299,-*)
Bestell-Nr. L6 85 12K Kassette
DM 29,90* (sFr. 24,90/öS 299,-*)

64'er-Ausgabe 11/85
Bestell-Nr. L6 85 11A
DM 29,90* (sFr. 24,90/öS 299,-*)

64'er-Ausgabe 10/85
Bestell-Nr. L6 85 10A
DM 29,90* (sFr. 24,90/öS 299,-*)

64'er-Ausgabe 9/85
Bestell-Nr. L6 85 09A
DM 29,90* (sFr. 24,90/öS 299,-*)

64'er-Ausgabe 8/85
Bestell-Nr. L6 85 08A
DM 29,90* (sFr. 24,90/öS 299,-*)

64'er-Ausgabe 7/85
Bestell-Nr. L6 85 07A
DM 29,90* (sFr. 24,90/öS 299,-*)

64'er-Ausgabe 6/85
Bestell-Nr. L6 85 06A
DM 29,90* (sFr. 24,90/öS 299,-*)

64'er-Ausgabe 5/85
Bestell-Nr. L6 85 05A
DM 29,90* (sFr. 24,90/öS 299,-*)

64'er-Ausgabe 4/85
Bestell-Nr. L6 85 04A
DM 29,90* (sFr. 24,90/öS 299,-*)

64'er-Ausgabe 3/85
Bestell-Nr. L6 85 03A
DM 29,90* (sFr. 24,90/öS 299,-*)

64'er-Ausgabe 2/85
Bestell-Nr. L6 85 02A
DM 29,90* (sFr. 24,90/öS 299,-*)

64'er-Ausgabe 1/85
Bestell-Nr. L6 85 01A
DM 29,90* (sFr. 24,90/öS 299,-*)

64'er-Sonderhefte

Sonderheft 7/86 – PEEKs & POKEs
Bestell-Nr. L6 86 S7D 1 Diskette
DM 29,90* (sFr. 24,90/öS 299,-*)

Sonderheft 6/86 – Grafik
2 Disketten mit allen Programmen
Bestell-Nr. L6 86 S6D1
DM 34,90* (sFr. 29,50/öS 349,-*)
1 Diskette mit Giga-CAD-Demos
Bestell-Nr. L6 86 S6D2
DM 19,90* (sFr. 17,-/öS 199,-*)
3 Disketten mit allen Programmen und Demos
Bestell-Nr. L6 86 S6D3
DM 49,80* (sFr. 43,50/öS 498,-*)

Sonderheft 5/86 – Grundwissen
Bestell-Nr. L6 86 S5D 1 Diskette
DM 29,90* (sFr. 24,90/öS 299,-*)

Sonderheft 4/86 – Abenteuer
Bestell-Nr. L6 86 S4D 2 Disketten
DM 34,90* (sFr. 29,50/öS 349,-*)

Sonderheft 3/86 – C16, C116, VC20, Plus/4
1 Diskette für VC20 und C16/116:
Bestell-Nr. L6 86 S3 CD
DM 29,90* (sFr. 24,90/öS 299,-*)
1 Kassette für VC20:
Bestell-Nr. L6 86 S3 KV
DM 19,90* (sFr. 17,-/öS 199,-*)
1 Kassette für C16:
Bestell-Nr. L6 86 S3 KC
DM 19,90* (sFr. 17,-/öS 199,-*)

Sonderheft 2/86 – Tips&Tricks
Bestell-Nr. L6 86 S2D Diskette
DM 29,90* (sFr. 24,90/öS 299,-*)

Sonderheft 1/86 – C128er
Bestell-Nr. L6 86 S1D Diskette
DM 29,90* (sFr. 24,90/öS 299,-*)

Sonderheft 8/85 – Assembler
Bestell-Nr. L6 85 S8D Diskette
DM 29,90* (sFr. 24,90/öS 299,-*)
Bestell-Nr. L6 85 S8K Kassette
DM 19,90* (sFr. 17,-/öS 199,-*)

Sonderheft 7/85 – Professionelle Anwendungen
Bestell-Nr. L6 85 S7D 2 Disketten
DM 34,90* (sFr. 29,50/öS 349,-*)
Bestell-Nr. L6 85 S7K 4 Kassetten
DM 34,90* (sFr. 29,50/öS 349,-*)

Sonderheft 6/85 – Top-Themen
Bestell-Nr. L6 85 S6 2 Disketten
DM 34,90* (sFr. 29,50/öS 349,-*)

Sonderheft 5/85 – Floppy, Datasette
Bestell-Nr. L6 85 S5D Diskette
DM 29,90* (sFr. 24,90/öS 299,-*)
Bestell-Nr. L6 85 S5K Kassette
DM 19,90* (sFr. 17,-/öS 199,-*)

Sonderheft 4/85 – Grafik
Bestell-Nr. L6 85 S4A
DM 29,90* (sFr. 24,90/öS 299,-*)

Sonderheft 3/85 – Spiele
Bestell-Nr. L6 85 S3 A 2 Disketten
DM 34,90* (sFr. 29,50/öS 349,-*)

Sonderheft 2/85 – Abenteuerspiele
Bestell-Nr. L6 85 S2
DM 34,90* (sFr. 29,50/öS 349,-*)

Sonderheft 1/85 – Tips&Tricks (2. überarb. Auflage)
Bestell-Nr. CB 023 Floppy-Utilities
DM 29,90* (sFr. 24,90/öS 299,-*)
Bestell-Nr. CB 024 Hilfsprogramme
DM 29,90* (sFr. 24,90/öS 299,-*)

* inkl. MwSt. Unverbindliche Preisempfehlung.

Bitte verwenden Sie für Ihre Bestellung und Überweisung die eingeheftete Postgiro-Zahlkarte, oder senden Sie uns einen Verrechnungs-Scheck mit Ihrer Bestellung. Sie erleichtern uns die Auftragsabwicklung, und dafür berechnen wir Ihnen keine Versandkosten.

INHALT

Vorwort

Aktuell

Grundlagen

Hardware

Software-Test

Anwendungen

Grafik-Listings

Spiele-Listings

Eingabehilfen

Tips&Tricks-Listings

Tips&Tricks

Mehr Dampf für C16 und Plus/4 mit der Floppy 1551

Es gibt sie zwar schon länger - so richtig auf den Markt kommt sie aber erst jetzt. Wir haben die Floppy 1551 von Commodore für Sie einmal unter die Lupe genommen.

Eine ziemlich lange Zeit haben sich die Gerüchte um die neue Floppystation von Commodore gehalten. Sie soll eine an den C16 und den Plus/4 angepaßte 1541 sein, die dabei allerdings bis zu viermal schneller ist.

Wir können Sie beruhigen. Die Gerüchte stimmen nur in einem Punkt und das ist die Geschwindigkeit. Die neue 1551 ist in der Tat viermal so schnell wie die 1541.

Um einen Umbau der 1541 handelt es sich dabei aber mit Sicherheit nicht. Das wird äußerlich schon durch einige besondere Merkmale deutlich. So ist die 1551 zum Beispiel schwarz und enthält ein Laufwerk mit Knebelverschluß. Anstatt der Buchsen für den seriellen Bus existiert nur ein Kabel, an dessen anderem Ende ein Kästchen mit Anschluß für den Erweiterungsport des C16 und Plus/4 angebracht ist.

Schraubt man das Gehäuse der 1551 auf, so fällt sofort die komplett neue Platine ins Auge. Diese Platine ist gegenüber der in der 1541 um fast die Hälfte kleiner. Sieht man genauer hin, so fallen auch gleich einige Details auf.

Da ist zuerst der Prozessor. Hier handelt es sich um einen 6510, der auch schon im C64 Verwendung findet. Der einzige Ein-/Ausgabebaustein, der auf der Platine vorhanden ist, ist ein TIA 6525. TIA steht dabei für »Triport Interface Adaptor«. Ein Baustein mit 3 x 8 Datenleitungen ersetzt also die beiden VIA 6522 in der 1541.

Wir haben es offensichtlich mit einer komplett neuen Elektronik zu tun, die natürlich auch ein entsprechend neues Betriebssystem (DOS) benötigt. Wie aus dem Handbuch zur 1551 ersichtlich, handelt es sich dabei aber zumindest um ein System, das befehlskompatibel zur »alten« 1541 ist.

Eine Sache fällt erst bei näherem Hinsehen auf. Richtet der Betrachter sein Augenmerk auf die Kabelverbindung zwischen Diskettenlaufwerk und Computer, so ist zu erkennen, daß es sich dabei um ein Parallelkabel handelt. Genau 16 Leitungen verbinden die 1551 mit dem C16 oder dem Plus/4, wodurch das Fehlen der Buchsen für den seriellen Bus erklärt ist.

Hinfällig werden damit jedoch auch eventuelle Hoffnungen auf den Anschluß der 1551 an den C64. Die Steckerausführung und auch -belegung am Expansionport des C16 ist vollkommen inkompatibel zum C64, so daß die 1551 als schnelle Alternative zur 1541 ausscheidet.

Ein vollkommen neues Gerät also, das zumindest für Besitzer eines C64 uninteressant zu sein scheint. Für den Anwender eines C16 oder eines Plus/4 stellt diese Diskettenstation aber sicherlich eine Alternative dar. Sie ist zwar nicht annähernd so schnell, wie es ihre Hardware erlaubt, die 1541 wird aber auf jeden Fall überholt.

Eine andere Entscheidungshilfe bekommt der Anwender aber auch von seiten der Kaufhof AG. Diese bietet den Commodore Plus/4 zusammen mit einer Diskettenstation 1551 für den sagenhaften Preis von 499 Mark an. Das dürfte die Gelegenheit für alle Einsteiger sein, denen ein Computersystem bisher zu teuer war. Immerhin kostet allein die 1551 im Handel normalerweise um 400 Mark, den Computer gibt es also fast gratis dazu (Bild)!

Für alle, die den Plus/4 vielleicht nicht kennen: Er ist weit über 90 Prozent kompatibel zum C16 (Basic-Programme und professionelle Software laufen auch auf dem Plus/4). Er besitzt 64 KByte RAM (der C16 nur 16 KByte), drei eingebaute Programme (Textverarbeitung, Tabellenkalkulation und Dateiverwaltung), einen Userport und eine angenehmere Tastatur als der C16. (ks)



Bild. Der Plus/4 und die Floppy 1551 stellen ein sehr leistungsfähiges und vor allem preiswertes Computersystem dar

Paintbox, ein Malprogramm für den C16

Zu den wenigen Anbietern von C16-Programmen gehört die deutsche Firma Kingsoft. Mit dem Programm Paintbox wird ein Programm geliefert, das trotz »speichersparender« Programmierung kaum Wünsche offenläßt.

Obwohl bei diesem Programm die Optionen nicht wie etwa beim Koalapainter oder Hi-Eddi+ über ein Grafikmenü ausgewählt werden können, läßt sich mit Paintbox sehr bequem und komfortabel malen. Zum Zeichnen läßt sich ebenso die Tastatur wie auch ein Joystick nutzen.

Im Menü stehen folgende Zeichenoptionen zur Verfügung:

Mit LINE lassen sich zwei Punkte in beliebiger Lage verbinden.

Linienketten können mit dem LINES-Befehl erzeugt werden.

Verwendet man den RAYS-Befehl, kann man eine Anzahl von Linien zeichnen, die alle von demselben Punkt ausgehen. Dadurch entstehen strahlenförmige Gebilde. Zur Konstruktion rechteckiger Umrandungen läßt sich die FRAME-Anweisung verwenden. Ausgefüllte Rechtecke, werden mit dem BOX-Befehl gezeichnet. Um das eigene Kunstwerk abzurunden, lassen sich mit dem Befehl CIRCLE und DISC Kreise sowie Scheiben darstellen. Umschlossene Flächen lassen sich mit dem PAINT-Befehl ausfüllen.

Mit den Befehlen COLOR und LUM lassen sich alle 121 Farbtöne erzeugen. Die Cursor-Option ermöglicht es dem Künstler, einen Pinsel aus acht verschiedenen Formen auszuwählen. Will man sehr genau zeichnen, so kann die Geschwindigkeit des Grafik-Cursors beliebig verlangsamt werden. Ist das Kunstwerk dann fertig, kann es nach Belieben auf Kassette oder Diskette gespeichert werden.

Paintbox läuft auf den Computern C16, 116 und Plus/4. Die 25 Mark für das Programm sind für Grafik-Fans mit Sicherheit eine lohnende Investition.

(Christian Q. Spitzner/hm)

Info: Kingsoft, Schnackebusch 4, 5106 Roetgen, 02408/5119

Synthesizer und Sequenzer: Music Master

Dieses Programm von Kingsoft gehört derzeit mit Sicherheit zu den besten Anwendungen für den C16. Das Programm entlockt dem »mageren« Tongenerator des C16/116 und Plus/4 die erstaunlichsten Sounds. Einem Vergleich mit dem C64 und C128 kann er aber trotzdem nicht standhalten.

Music Master ist in zwei Teile gegliedert, die untereinander verknüpft werden können.

Teil 1 ist ein Synthesizer, der sich über die Tastatur spielen läßt. Die dazugehörige Grafik ist recht nett ausgefallen. Es wird ein Teil einer Klaviertastatur nachgebildet, auf der die gerade gedrückte Taste angezeigt wird. Die restlichen Tasten dienen zur Steuerung des Sounds.

Der Synthesizer verfügt über zwei Stimmen. Verknüpft man beide, so läßt sich auch im Zweiton-Akkord spielen. Wenn zusätzlich die Stimmen untereinander leicht verstimmt werden, so entsteht eine Resonanzfrequenz und der Ton wirkt voller.

Durch Veränderung der Hüllkurve lassen sich viele unterschiedliche Effekte erzielen. Viele Instrumente können durch Variation der Hüllkurve nachgeahmt werden.

Der zweite Teil von Music Master ist ein Sequenzer, der kaum noch Wünsche offenläßt. Beide Stimmen können unabhängig voneinander gesteuert werden. Will man ein Lied eingeben, so stehen dem Spieler zwei Möglichkeiten zur Verfü-

gung. Zum einen kann das Musikstück in Echtzeit gespielt werden. Zum anderen lassen sich die Noten einzeln eingeben. Die Noten können sehr leicht editiert werden. Es lassen sich ganze Teile innerhalb einer Stimme, aber auch in die andere Stimme kopieren. Es können Töne eingefügt und Teile des Musikstückes versetzt werden. Da beide Stimmen unterschiedlich programmiert werden können, lassen sich tolle Musikstücke spielen. Dabei könnte zum Beispiel Stimme 1 die Melodie übernehmen, während Stimme 2 Schlagzeug oder Baß spielt.

Für Leute, die lieber am Synthesizer in »Echtzeit« spielen, wurden für Stimme 2 zusätzlich 10 Begleitrhythmen integriert. Ein zusätzliches Begleitmuster kann selbst programmiert werden.

Was der Music Master kann, läßt sich am besten am mitgelieferten Demo-Lied sehen (hören). Dieses Musikstück zeigt, was wirklich im Tongenerator des C16 steckt. Übrigens kann ein erstelltes Musikstück auch ins eigene Programm übernommen werden.

Das Programm ist auf Kassette für 29 Mark erhältlich.

(Christian Q. Spitzner/hm)

Info: Kingsoft, Schnackebusch 4, 5106 Roetgen, 02408/5119

C16/116-Programmsammlung: 11 Top-Programme für den C16

Von Markt&Technik wird eine Programmsammlung, bestehend aus 11 Programmen für den C16/116 und Plus/4, angeboten. Sie ist auch in Kaufhäusern erhältlich.

Die Disketten enthalten viele Spiele, eine kleine Textverarbeitung und sogar eine Basic-Erweiterung.

Zum Lieferumfang der Kassetten (Preis 29,90 Mark) gehört auch ein Handbuch mit ausführlicher Beschreibung der Programme.

Folgende 11 Programme befinden sich auf den Kassetten:

Robot 3.5

Der Spieler muß in einem Labyrinth nach einem Schlüssel suchen und unterwegs Schätze aufsammeln. Dabei machen ihm aber Roboter das Leben schwer.

Eine Berührung der Roboter sowie der elektrischen Wände ist für den Spieler äußerst ungesund.

Wurmi

Der Spieler steuert einen ununterbrochen laufenden Wurm auf dem Bildschirm und sammelt dabei Äpfel ein. Der Wurm muß aufpassen, daß er nicht gegen eine Wand fährt, beziehungsweise sich selbst »in den Schwanz beißt«. Hier spielt auch die Zeit eine Rolle. Werden die Äpfel nicht schnell genug verspeist, muß der Wurm leider an »Unterernährung« sterben.

Datagenerator C16

Das Programm »Datagenerator C16« wandelt einen frei zu wählenden Speicherbereich in DATA-Zeilen um. Der Datagenerator wird anschließend gelöscht. Eine passende Einleseschleife wird leider nicht generiert.

Basic-Tool

Diese Basic-Erweiterung ergänzt das Basic 3.5 des C16 um 13 zusätzliche Basic-Befehle. Die neuen Befehle werden im Programm in Tokens umgewandelt und benötigen deshalb nicht mehr Speicher als ein normaler Basic-Befehl. Die Erweiterung umfaßt folgende Befehle:

Mit WINDOW kann durch Angabe der Koordinaten ein Fenster definiert werden. RENEW macht ein versehentliches Löschen eines Basic-Programms rückgängig. Die Befehle DMERGE und MERGE verknüpfen zwei Programme. Das 2. Programm wird von Diskette oder Kassette eingelesen und muß größere Zeilennummern haben als das erste Programm.

Mit der SET-Anweisung kann der Cursor an einer beliebigen Stelle positioniert werden. Soll der Zeichensatz verändert werden, so kann dieser mit ZTRANS in den RAM-Bereich kopiert werden. ZON setzt die Zeiger für den Zeichensatz auf eine beliebige Adresse. Die MULTI-Anweisung schaltet den Multi-Color-Modus ein. Mit dem Befehl ZNORM wird der Mehrfarbenmodus ausgeschaltet, und der Zeichensatz wieder aus dem ROM gelesen. Mit PUT kann an einer beliebigen Stelle am Bildschirm ein Zeichen ausgegeben werden. FAST schaltet den Bildschirm ab, wodurch der C16 um den Faktor 1,5 schneller wird. Der Normalzustand kann mit dem SLOW-Befehl wiederhergestellt werden. Der 13. und letzte Befehl (OFF) schaltet die Erweiterung aus.

Kniffel

Bekanntes Würfelspiel für bis zu vier Spieler. Ein Spiel gegen den Computer ist nicht möglich.

Poker

Das Spiel Poker simuliert eine Pokerrunde mit fünf Personen. Der Computer stellt vier Personen, während der Spieler als fünfte Person gegen den Computer antreten kann. Jeder Spieler kann setzen, erhöhen oder Karten tauschen.

Diabolo

Ein hervorragendes Geschicklichkeitsspiel mit toller Grafik. Der Spieler hat die Aufgabe, auf einem fernen Planeten Roboter-Maschinen zu zerstören und gefangene Menschen zu befreien. Es ist darauf zu achten, daß weder Sie, noch die befreiten Menschen, noch Ihre eigene Basis von Robotern überrollt werden. In jedem Falle muß der Spieler sein Leben lassen. Das Spiel kann in sechs verschiedenen Spielstufen gespielt werden.

Bridge

Bridge ist ein Geschicklichkeitsspiel, bei dem es darum geht, Brückenteile vor einem sich nähernden Auto abzuwerfen. Mißglückt die Aktion dreimal, muß Jonny Car jämmerlich im reißenden Fluß ertrinken.

No Exit

In No Exit muß der Spieler versuchen, in einem Labyrinth den Ausgang zu finden. Bevor man jedoch aufgibt, kann man den Irrgarten auch aus der Vogelperspektive betrachten, was natürlich Zeit und Punkte kostet.

Text 1.0

Text 1.0 ist ein kleines Textverarbeitungsprogramm, mit dem sich kurze Notizen verarbeiten lassen. Das Programm ist menügesteuert und hat folgende Optionen zur Auswahl:

Laden: Texte werden von Datasette gelesen.

Speichern: Der sich im Textspeicher befindliche Text wird auf Band gesichert.

Drucken: Text wird aufs Papier gebracht.

Formatfestlegung: Dient zur Einstellung der Ränder und Tabulatoren.

Anfügen: Weitere Eingaben werden an den Text angefügt.

Neueingabe: Der Textspeicher wird gelöscht. Die Eingabe beginnt bei Zeile 1.

Suchen: Hilft bei der Suche nach bestimmten Wörtern im Text.

Es können im Text zusätzlich Steuerbefehle eingesetzt werden, die bei der Formatierung des Textes helfen. Einem Vergleich mit dem Text-Manager kann das Programm jedoch nicht standhalten.

Cannon

Ein Schießspiel, bei dem es darum geht, den Feind auf der anderen Seite eines Hügels mit einer Kanone zu treffen. Dabei spielen Windgeschwindigkeit, Hindernishöhe, Abschußwinkel und Geschwindigkeit eine große Rolle. Der Feind läßt sich jedoch alles gefallen. Er schießt nicht zurück und wartet darauf, von Ihnen getroffen zu werden.

Unter diesen elf Programmen befindet sich bestimmt für jeden das passende Programm. Für 29.90 Mark sind 2 Kassetten mit ausführlicher Dokumentation ein angemessen fairer Preis.

(Christian Q. Spitzner/hm)

Info: Markt&Technik-Verlag, Hans-Pinsel-Str. 2, 8013 Haar, 089/4613-0

Speichererweiterung wird billiger

Im letzten C16-Sonderheft haben wir die Speichererweiterung von Kingsoft vorgestellt und getestet. Mittlerweile ist der Preis für die 64-KByte-Erweiterung erheblich gesunken. Zur Zeit ist sie für 139 Mark zu erhalten. (kn)

Info: Kingsoft, Schnackebusch 4, 5106 Roetgen, 02408/5119

Speichererweiterung einbauen lassen

In diesem Sonderheft beschreiben wir in dem Artikel »Aus klein macht groß«, wie Sie eine Speichererweiterung auf 64 KByte selbst bauen können. Sollten Sie nicht zu den versierten Hardware-Bastlern gehören, so können Sie den Umbau gegen 100 Mark Vorkasse bei Elektronik-Technik und bei Manfred Velt in Auftrag geben. Schicken Sie dazu den Computer in einer stoßsicheren Verpackung an eine der beiden nachstehenden Adressen. (kn)

Elektronik-Technik U. Peters, Tannenweg 9, 2351 Trappenkamp, 04323/3991
Manfred Velt, Albrecht-Dürer-Str. 26, 8540 Schwabach

64 KByte im Selbstbau

Die Firma SAS bietet eine Speicherplatine für den C16 an. Als Bausatz kostet sie 97 Mark, als Fertiggerät 144 Mark. (kn)

SAS Hermann-Josef Bernd, Langgasse 93, 5216 Niederkassel 5, 0228/452626

Fragen und Antworten zum C16

Deutsche Sonderzeichen

Gibt es eine Möglichkeit zur Darstellung deutscher Sonderzeichen (ä,ö,ü,Ä,Ö,Ü,ß) auf dem C16? Holger Sternberg

Ja, es gibt die Möglichkeit deutsche Sonderzeichen zu erzeugen. Dazu muß jedoch der Zeichensatz des C16 in den RAM-Speicher kopiert werden. Dort können dann alle Zeichen beliebig verändert werden. Dies geht dann natürlich auf Kosten einiger Sonderzeichen. Es können also keine neuen Zeichen hinzugefügt werden, sondern höchstens bestehende Zeichen verändert werden.

C64/VC 20-Programme auf dem C16

Sind Programme, die auf dem C64 oder VC 20 geschrieben wurden, auch auf dem C16 zu verwenden?
Markus Müller

C64/VC 20-Programme laufen problemlos auf dem C16, wenn folgende Bedingungen erfüllt sind:

- Es muß ein Basic-Programm sein (nicht Simons-Basic oder ähnliches).
- Das Programm darf keine Maschinensprache-Anteile mit DATA-Zeilen enthalten.
- Das Programm darf keine PEEK-, POKE-, SYS- und WAIT-Befehle enthalten.
- Die Speicherkapazität des C16 muß für das Programm ausreichen.

Handelt es sich bei den POKEs allerdings nur um einfache Dinge, wie beispielsweise Bildschirmfarben einstellen, so können Sie diese getrost weglassen und anschließend Ihr Glück versuchen.

Speichererweiterung für den C16

Ich besitze seit Weihnachten einen C16. Seit einiger Zeit allerdings habe ich Schwierigkeiten damit, weil der Speicher des C16 nicht so viel Kapazität aufweist. Meine Frage ist nun an Sie, wie man diese Speicherkapazität erweitern kann.
Claudia Bacherler

Es gibt Speichererweiterungen für den C16. Von Kingsoft wird eine 16-KByte- sowie eine 64-KByte-Speichererweiterung angeboten. Ein Testbericht erschien im Sonderheft 3/86 unter der Überschrift »60671 BYTES FREE beim C16«. Weitere Informationen finden Sie unter der Rubrik »Neue Produkte« in diesem Heft.

C64-Kassetten für den C16

Wegen des besseren Basic (Basic 3.5) habe ich mir vor kurzem einen C16 mit Datasette 1531 zugelegt. Ich habe jetzt das Problem, daß ich meine Programme, die ich mit dem C64 auf Kassette gespeichert habe, nicht mehr laden kann. Bei einem Lade-Versuch tut mein C16 so, als wäre die Kassette leer. Welchen POKE muß ich verwenden, daß ich die C64-Kassetten in meinen C16 laden kann?
Robert Wilsperger

Es gibt leider keinen »POKE«, mit dem man so einfach C64-Kassetten laden kann. Das Problem besteht darin, daß alle Commodore-Modelle eine andere Aufzeichnungsgeschwindigkeit haben. Deshalb ist es ebenso unmöglich, VC 20-Kassetten mit dem C64 oder C16 zu lesen. Die einzige Möglichkeit, Ihre Programme in den C16 zu bekommen, wäre der Umweg über ein Floppy-Laufwerk. Sie müßten alle Ihre Programme auf Diskette speichern und könnten diese mit dem C16 wieder laden. An beiden Computern kann das gleiche Laufwerk (1541) angeschlossen werden.

LISTschutz für den C16

Gibt es für den C16 einen LISTschutz und einen Kopierschutz? Günther Ecken

Ja. Es gibt mehrere Möglichkeiten des LISTschutz: Mit POKE 775,128 werden bei einem LIST-Versuch nur die Zeilennummern ausgegeben. Das Programm selbst bleibt unsichtbar. Eine andere Möglichkeit besteht durch POKE 775,252. Versucht man hier, das Programm zu listen, wird ein Reset ausgeführt.

Einen Kopierschutz gibt es in der Regel nur auf Disketten oder Kassetten. Ein eigenes Programm zu schützen ist normalerweise nicht so einfach. Es gibt jedoch eine einfache Möglichkeit, ein Programm vom Speichern auf Kassette zu schützen. Geben Sie folgende drei Befehle ein:

POKE 1525,0
POKE 1526,0
POKE 1527,0

Der C16 simuliert den Save- und Verify-Vorgang und alles sieht aus, als wäre es in Ordnung.

<STOP/RESTORE> beim C16?

Ich besitze seit einiger Zeit den C16. Vorher hatte ich einen VC 20. Wenn sich mein VC 20 in einem Maschinenprogramm verfangen hat oder in einer falschen WAIT-Anweisung hängengeblieben ist, konnte ich ihn mit <STOP/RESTORE> immer »zurückholen«. Auf meinem C16 finde ich leider keine <RESTORE>-Taste mehr. Gibt es einen Ersatz? Fritz Mayr

In der Tat gibt es einen Ersatz. Ist Ihr C16 abgestürzt, so läßt er sich meist mit einem Trick wieder zum Leben erwecken. Drükken Sie die <RUN/STOP>-Taste und gleichzeitig den Reset-Taster. Lassen Sie die <RUN/STOP>-Taste erst dann los, wenn das Anfangsbild des Monitors auf dem Bildschirm erscheint. Befinden Sie sich im Monitor, so können Sie diesen in der Regel mit X (<RETURN>) verlassen. Das Basic-Programm befindet sich immer noch im Speicher. Vorauszusetzen ist natürlich, daß keine wichtigen Systemadressen zerstört wurden.

Plus/4-Programme auf C16?

Sind Programme für den Plus/4 auch auf dem C16 mit 64-KByte-Erweiterung lauffähig? Tobias Rademacher

Diese Frage läßt sich normalerweise mit »ja« beantworten. Kleine Einschränkungen sind allerdings notwendig. Die im Plus/4 vorhandene RS232-Schnittstelle und die integrierte Software darf von dem Programm nicht genutzt werden.

Weitere Informationen dazu erhalten Sie in dem Artikel »Drei 'ungleiche' Brüder« in diesem Heft.

C64-ROM im C16

Ist es möglich, durch Austauschen der ROM-Bausteine im C16/C116 mit EPROMs (Betriebssystem des C64) den C16 für C64-Programme lauffähig zu machen?
Thomas Tiemann

Auf diese Weise können Sie C64-Programme auf dem C16 nicht zum Laufen bringen. Ganz im Gegenteil, der C16 würde nach dieser Änderung gar nicht mehr funktionieren. Ein Grund dafür ist der TED des C16, der im C64 nicht vorhanden ist. Die CIAs, der Sound- und der Videochip des C64 sind im C16 im TED zusammengefaßt.

Programme für C16

Es gibt kaum Spielelistings und Anwendungsprogramme für den C16/116 (Textverarbeitung oder Matheprogramme). Kennen Sie Hersteller?
Dirk Koch

Zunächst möchten wir Sie noch auf unser C16-Sonderheft (3/86 und 8/86) aufmerksam machen.

Ferner wird ab Juli '86 ein C16-Programmpaket von Markt &Technik mit zwei Kassetten in den Kaufhäusern erhältlich sein. Viele Spiele und Anwendungen sind darauf enthalten. Außerdem: Immer mehr Software-Hersteller bieten jetzt Programme für den C16 an. Sehen Sie sich einmal die Anzeigen im 64'er-Magazin an.

Profi-Drucker für C64/C128!

Interface Inclusive!

Funktion garantiert!

Zum Super-Paket-Preis!

- Druckerpaket: Drucker + Interface sofort funktionsfähig
- Druckwegoptimiert
- voll grafikfähig

Speedy 100-80

100 cps Matrix
25 cps NLQ, gute Schriftqualität
d. Karbon farbband

Paketpreis:
DM 1.138,86*
(DM 999,– + MwSt.)

Juki 6100



22 cps Typenraddrucker
Superschriftqualität – Karbonfarbband – große Auswahl an Typenrädern (TA-kompatibel)

Paketpreis:
ab DM 1.594,86*
(DM 1.399,– + MwSt.)

Fujitsu DX2100/2200

220 cps Matrix
mit 80 o. 136 Spalten
44 cps NLQ,
auf Farbe nachrüstbar

Paketpreis:
DX2100
ab DM 2.046,30*
(DM 1.795,– + MwSt.)
DX2200
ab DM 2.506,86*
(DM 2.199,– + MwSt.)

Fujitsu DPMG9



180 cps Matrix
25 cps NLQ, robust und zuverlässig

Paketpreis:
DM 1.708,86*
(DM 1.499,– + MwSt.)

Siemens PT 88

150 cps Tintenstrahldrucker unter 45 dB(A)

Paketpreis:
DM 3.076,86*
(DM 2.699,– + MwSt.)

*unverbindl. Preisempfehlung

Bitte senden Sie mir Unterlagen über:

☐ Speedy 100-80 ☐ DPMG9 ☐ Juki 6100
☐ DX2100/2200 ☐ Siemens

Firma ____ Name ____

Anschrift ____

Telefon ____

64'er SH 8/86

MACROTRON



Stahlgruberring 28 · 8000 München 82 · Tel. (0 89) 42 08-0
Tx. 529 448 mato · Teltex 897 280 = mato · Tfax 089-429 563

MACROTRON Peripherie-Fachhändler:

1000 Berlin: Thor Text, Tel. 0 30/24 60 90 • Microcomputer Laden, Tel. 0 30/8 82 65 91 • Systemhaus Saar KG, Tel. 0 30/8 91 80 82 • Jörg Korb Elektronik, Tel. 0 30/2 62 74 75 • Ehrig GmbH, Tel. 0 30/3 41 70 21 • Alpha Computers GmbH, Tel. 0 30 / 8 91 10 82 • Comco, Tel. 0 30/3 23 86 91 • DWF Luther, Tel. 0 30/8 15 94 46 • **2000** Hamburg: Computer Contor, Tel. 0 40/23 46 27 • Kreling & Partner, Tel. 0 40/2 51 20 39 • **2080** Pinneberg: GDS, Tel. 0 41 01/2 37 99 • **2300** Kiel: Computer Partner, Tel. 04 31/9 50 25 • Pabst & Schmitt OHG, Tel. 04 31/68 11 11 • **2350** Neumünster: Bernhardt Microcomputersysteme, Tel. 0 43 21/35 89 • **2400** Lübeck: Nielandt Computerservice, Tel. 04 51/62 38 51 • **2800** Bremen: Comdat GmbH, Tel. 04 21/32 04 09 • **2820** Bremen: AD Computertechnik, Tel. 04 21/6 36 12 68 • **2870** Delmenhorst: Werner Büro- und Datentechnik, Tel. 0 42 21/1 44 70 • **2940** Sande: Müller Büroorganisation, Tel. 0 44 22/42 42 • **3005** Hemmingen: DP Elektronik, Tel. 05 11/4 20 00 92 • **3055** Hagenburg: W. Busse Computer und Elektronik, Tel. 0 50 33/76 77 • **3057** Neustadt: Volker Hellborg Labor f. Funk- und Meßtechnik, Tel. 0 50 32/6 27 96 • **3300** Braunschweig: Apel Bürocenter GmbH, Tel. 05 31/7 30 71 • **3400** Göttingen: WK Elektronik, Tel. 05 51/9 60 61 • **3400** Göttingen: RIS Software, Tel. 05 51/9 62 82 • **3500** Kassel: Gefad GmbH, Tel. 05 61/77 10 71 • **3501** Fuldabrück: Rühlig Bürosysteme, Tel. 05 61/58 20 57 • **4000** Düsseldorf: Max Lips, Tel. 02 11/49 90 52 • **4050** Mönchengladbach: RFI GmbH, Tel. 0 21 61/6 00 60 • **4100** Duisburg: Helmut Rennen GmbH, Tel. 02 03/2 49 26 • Kaeppke & Krämer, Tel. 02 03/77 34 25 • **4300** Essen: Helmut Rennen GmbH, Tel. 02 01/23 71 39 • B.T.O. Büro-Technik, Tel. 02 01/77 60 01 • **4400** Münster: Software City, Tel. 02 51/4 08 66 • **4500** Osnabrück: HDS Computer GmbH, Tel. 05 41/6 80 18 • PCO, Tel. 05 41/66 50 • **4690** Herne: Groß Computersysteme • **4800** Bielefeld: Infotext, Tel. 05 21/3 40 22 • **4900** Herford: Rezo GmbH, • **4995** Sternwede: HGS Computer GmbH, Tel. 0 57 45/25 55 • **5000** Köln: Klicman Elektronik, Tel. 02 21/24 12 23 • Grosse, Tel. 02 21/5 99 19 45 • Büroelektronik, Tel. 02 21/54 43 06 • **5090** Leverkusen: Rolf Rocke Computer, Tel. 0 21 71/26 24 • **5100** Aachen: ICT, Tel. 02 41/1 20 59 • **5204** Lohmar: Datasoft GmbH, Tel. 0 22 46/81 77 • **5300** Bonn: HS Datentechnik GmbH, Tel. 02 28/25 20 91 • TBS-Informations-Systeme, Tel. 02 28/23 20 45 • **5352** Zülpich-Ülpenich: Computer Zens, Tel. 0 22 52/31 84 • **5400** Koblenz: Wir Bürosyst. GmbH, Tel. 02 61/80 20 14 • **5620** Velbert: hsn Nußbaumer, Tel. 0 20 52/8 10 63 • **5630** Remscheid: Comsoft, Tel. 0 21 91/2 10 33 • **5650** Solingen: Conex, Tel. 02 12/7 54 49 • **5788** Winterberg: Karl Groß, Tel. 0 29 81/77 76 • **5800** Hagen: Bröker Computertechn., Tel. 0 23 31/6 63 16 • **5810** Witten: Computer Jagusch, Tel. 0 23 02/5 20 46-47 • **6000** Frankfurt: Utimaco Software GmbH, Tel. 0 69/57 20 45 • **6100** Darmstadt: Max Lips GmbH, Tel. 0 61 51/2 63 43 • **6200** Wiesbaden: Everyware Computer GmbH, Tel. 0 61 21/44 90 67 • **6230** Frankfurt 80: MCT, Tel. 0 69/37 32 77 • **6231** Sulzbach: STS-Spectrum, Tel. 0 61 96/2 90 71 • **6250** Limburg: wir Bürosysteme GmbH, Tel. 0 64 31/2 60 88 • **6300** Gießen: Ordat GmbH & Co., Tel. 06 41/79 41-56 • **6600** Saarbrücken: Schommer GmbH, Tel. 06 81/58 18 32 • **6710** Frankenthal: Movecro GmbH, Tel. 0 62 33/1 33 39 • **6720** Speyer: Dörr Computer, Tel. 0 62 32/7 76 21 • **6730** Neustadt: ICR GmbH, Tel. 0 63 27/3 90 • **6758** Lauterecken: Schwenk, Tel. 0 63 82/62 66 • **6800** Mannheim: Webidata GmbH, Tel. 06 21/37 10 34 • IK-Systeme, Tel. 06 21/79 70 08 • **6835** Brühl: Dobbertin, Tel. 0 62 02/7 14 17 • **6922** Meckesheim: E. Geißler Micro, Tel. 0 62 26/32 68 • **6940** Weinheim: HMR GmbH, Tel. 0 62 01/1 20 37 • **6973** Boxberg: CD-Comp.-Dienst, Tel. 0 79 30/20 71 • **7000** Stuttgart: Bierbrauer & Nagel • BWI-Stratos GmbH, Tel. 07 11/22 42 38 • Drost & Siegle, Tel. 07 11/81 44 88 • Computerland, Tel. 07 11/29 44 18 • TUD GmbH, Tel. 07 11/7 15 68 69 • Konrad Westermeier, Tel. 07 11/61 20 84 • Kunkel Industriebedarf, Tel. 07 11/88 47 11 • Rausch & Partner, Tel. 07 11/88 19 92 • **7024** Filderstadt: Datapec, Tel. 07 11/77 50 45 • **7030** Böblingen: P + B Abele, Tel. 0 70 31/6 20 50 • **7050** Waiblingen: P. Nikola GmbH, Tel. 0 71 51/5 50 69 • **7146** Tamm/Hohenstange: Computerbüro Fischer, Tel. 0 71 41/60 30 91 • **7300** Esslingen: Grässer Computersyst., Tel. 07 11/3 16 17 86 • **7400** Tübingen: Computer Point, Tel. 0 70 71/2 43 39 • **7417** Pfullingen: Dr. P. Gehrmann, Tel. 0 71 21/7 40 76 • **7440** Nürtingen: Bürotex GmbH, Tel. 0 70 22/3 52 31 • **7500** Karlsruhe: gds GmbH, Tel. 07 21/49 35 20 • Krempel & Klingel, Tel. 07 21/2 16 78 • **7613** Hausach: Streit Datentechnik, Tel. 0 78 31/80 20 • **7707** Eugen: Hartmut Schaber, Tel. 0 77 33/84 01 • **7730** VS-Weilersbach: Maier Datensysteme, Tel. 0 77 21/7 03 22 • **7730** VS-Schwenningen: BUS GmbH, Tel. 0 77 20/3 80 71 • **7750** Konstanz: Rösler Computertechnik, Tel. 0 75 31/2 18 32 • **7798** Pfullendorf: ICR PV GmbH, Tel. 0 75 52/16 81 • **7800** Freiburg: Parsch Electronic, Tel. 07 61/44 20 94 • Fritz Computer, Tel. 07 61/70 04 04 • Computer Partner, Tel. 07 61/7 80 44 • **7900** Ulm-Böfingen: Interplan TBS Software, Tel. 07 31/2 69 49 • C.T.O. GmbH, Tel. 07 31/2 61 07 • **8000** München: Böwe Systemvertrieb, Tel. 0 89/50 33 93 • Abacus Computercenter, Tel. 0 89/28 60 81 • Servonic, Tel. 0 89/48 32 53 • Bürop, Tel. 0 89/1 78 30 34 • Stitz Computer GmbH, Tel. 0 89/93 10 65 • **8031** Gilching: CPV, Tel. 0 81 05/2 37 03 • **8057** Eching: Bahe Bürosysteme GmbH, Tel. 0 89/3 19 40 75 • **8110** Hofheim: Büroorg. Schnabel, Tel. 0 88 47/66 11 • **8120** Weilheim: Computerstudio Hutter, Tel. 08 81/12 23 • **8202** Bad Aibling: Hieble + Kunze, Tel. 0 80 61/60 17 • **8228** Freilassing: CF-Computer Funktion, Tel. 0 86 54/6 41 56 • **8300** Landshut: Computerstudio Landshut, Tel. 08 71/2 82 75 • **8391** Mauth: Gibis-EDV, Tel. 0 85 57/410 • **8510** Ursensoll: CPV GmbH, Tel. 0 81 05/14 95 • **8580** Bayreuth: Unidata, Tel. 09 21/5 66 88 • **8630** Coburg: Beyer Computer, Tel. 0 95 61/97 56 • **8700** Würzburg: August Fassnacht, Tel. 09 31/2 00 15-0 • **8780** Gemünden: EDV Beratung Reuter, Tel. 0 93 51/26 27 • **8880** Dillingen: Reitzner Bürozentrum, Tel. 0 90 71/20 60 • **8900** Augsburg: C & S Computer, Tel. 08 21/51 51 20 • Ing. Büro Bartholomäus, Tel. 08 21/31 19 11 • Böwe Systemvertrieb, Tel. 08 21/5 70 25 72 • **8912** Kaufering: ISK GmbH, Tel. 0 81 91/7 03 02 • **8920** Memmingen: Computerladen, Tel. 0 83 31/59 42



Der C 16 hat gegenüber seinem »großen« Bruder, dem C 64, zwar deutlich weniger Speicherplatz, aber dafür ein sehr umfassendes Basic, das der C 64 nicht zu bieten hat. Hier zeigen wir Ihnen die Funktionen von häufig gebrauchten Basic-Befehlen. Bauen Sie sich mit Hilfe dieses Kurses nach und nach Ihr eigenes Programm auf.

Das folgende kleine Inhaltsverzeichnis soll Ihnen einen Überblick über diesen Basic-Kurs geben.

Eine ganze Zeit lang galt der C 16 als das schwarze Schaf in der Familie der Heimcomputer von Commodore. Der Speicher ist klein, die akustischen Eigenschaften sind begrenzt, er hat keine Sprites – das sind präzise selbstdefinierbare Figuren. Auch an den Schnittstellen, das sind die Verbindungen des Computers mit der Außenwelt, wurde gespart.

Nicht gespart worden dagegen ist bei der eingebauten Programmiersprache Basic.

Der alte VC 20 und der Dauerbrenner C 64 weisen 69 verschiedene Befehle auf; der C 16 dagegen ist mit 120 Befehlen ausgerüstet! Nur der große C 128 hat noch mehr Befehle, nämlich 164.

Besonders dieser Befehlsvorrat, sicher aber auch der niedrige Preis, hat dem C 16 in der letzten Zeit zu einem rasanten Aufstieg verholfen. Auch der kleine Speicher ist inzwischen erweiterbar – auf 60600 verfügbare Byte! Damit übertrumpft der C 16 in drei wesentlichen Punkten den C 64: Basic-Befehle, Speicher, Preis.

1. Der Lerncomputer

Ist der C 16 wirklich ein echter Lerncomputer?

Ein Computer, der geeignet sein soll, Anfängern das Verständnis und den Gebrauch zu erleichtern, muß drei Voraussetzungen erfüllen:
a) er muß einfach zu bedienen sein
b) er muß eine simple Programmiersprache besitzen, die aber aus vielen Befehlen bestehen soll, so daß später auch komplizierte Programme geschrieben werden können
c) es muß genügend und leicht verständliche Literatur geben, die bei Anfängern die Begeisterung zur Computerei weckt.

Die ersten beiden Bedingungen sind beim C 16 ganz sicher gegeben.

Daß aber die dritte Bedingung zutrifft, kann man mit gutem Gewissen heute noch nicht behaupten. Einen Teil dieser Lücke zu füllen ist das Ziel dieses Kurses.

Ich werde allerdings nicht von ganz vorn anfangen. Im Sonderheft 5/86, das im Mai erschienen ist, steht von Seite 40 bis Seite 68 ein »Basic-Kurs von Anfang an«. Er ist zwar ursprünglich für den C 64 geschrieben worden, aber am Ende hat sich herausgestellt, daß er für den C 16 genauso anwendbar ist.

Dieser Anfangskurs schafft nämlich mit der Besprechung von 23 grundlegenden Basic-Befehlen eine Basis, die auch für den C 16 gilt. Ich brauche diese Befehle deshalb hier nicht zu wiederholen. Wenn Sie gerade erst Ihren C 16 bekommen und noch nie programmiert haben, dann ist der Anfangskurs im Sonderheft 5/86 genau der richtige Einstieg für Sie. Selbst wenn Sie schon eine Ahnung vom Programmieren haben, ist die Lektüre des Anfangskurses sicher keine Zeit-

verschwendung, da er viele Tips und Kochrezepte zu den folgenden Befehlen enthält:

PRINT	IF-THEN	CHR$	FOR-TO
LET	END	ASC	STEP
LIST	REM	LEFT$	NEXT
RUN	RND(X)	MID$	READ
NEW	INT(A)	RIGHT$	DATA
INPUT	GET	LEN	RESTORE
GOTO			

Dieser Kurs schließt sich also nahtlos an den Anfangskurs an und geht in gemischter Folge sowohl auf weitere Basic-Befehle ein, die in allen Commodore-Computern enthalten sind, als auch auf die speziellen C16-Befehle. Ich möchte allerdings darauf hinweisen, daß diese speziellen Befehle des C16 auch im C128 enthalten sind. Zur besseren Übersicht, wie sich die Befehle auf die Commodore-Computer verteilen, habe ich sie im Anhang dieses Kurses in einer Tabelle zusammengestellt.

Ich mache übrigens keinen Unterschied zwischen Befehlen, Anweisungen, Funktionen und dergleichen. Die Unterschiede sind oft nur akademisch und haben auf die Anwendung keinen Einfluß. Für Sie ist es sicher einfacher, wenn ich alle »Befehle« nenne.

Bevor wir anfangen, will ich schnell noch die Methode erwähnen, mit der Sie den Kurs am besten verfolgen und den Stoff nachvollziehen können.

Ich habe diese Vorgehensweise »Simulationsmethode« genannt, was nichts anderes bedeutet, als den Basic-Kurs vor dem Computer sitzend zu lesen und alle Listings sofort einzutippen und auszuprobieren.

Anmerkung:
Weitere Informationen über den C16 finden Sie im 64'er-Sonderheft 3/1986, das speziell dem C16 und dem VC 20 gewidmet ist.
Auch in jeder Ausgabe des 64'er-Stammheftes gibt es immer spezielle Tips und Tricks für den C16.

2. Die kleinen Unterschiede des C16

Ich habe einige Absätze weiter oben behauptet, der Inhalt des Anfangskurses gelte für alle Commodore-Computer. Das war etwas voreilig, denn ein paar kleine Unterschiede gibt es doch, allerdings nicht die Basic-Befehle betreffend.

Die kleinen Unterschiede betreffen die Tastatur des C16, speziell die Steuertasten und die Funktionstasten:
- die Taste <ESC> (links oben) ist neu hinzugekommen. Ihre Funktion ist im Handbuch beschrieben.
- die Funktionstasten (rechts), die beim VC 20 und C64 nur über ihre ASCII-Codes verwendet werden können, sind beim C16 mit Funktionen belegt. Wie die im Anfangskurs auf Seite 61 bis 63 beschriebene Verwendung der Funktionstasten auch beim C16 funktioniert, werde ich noch näher erklären.
- es gibt keine <RESTORE>-Taste. Ihre Funktion wird durch den viel radikaleren Reset-Taster ersetzt.
- alle anderen Steuertasten arbeiten so, wie im Anfangskurs beschrieben; einige von ihnen befinden sich auf der Tastatur des C16 an einer anderen Stelle, was uns aber nicht stören soll.

3. Variable machen Programme variabel

Variablen habe ich bereits im Anfangskurs behandelt, allerdings nur in einer ersten »Ausbaustufe«. Bevor ich ihre Bedeutung erweiternd erkläre, will ich – ihrer Wichtigkeit wegen – das im Anfangskurs Gesagte kurz wiederholen.

Mit dem Begriff »Variable« werden Namen bezeichnet, unter denen sich der Computer Zahlenwerte oder Buchstabenfolgen merken kann. »Merken« bedeutet bei einem Computer, daß die Zahlen oder Buchstaben im Speicher festgehalten sind und, so oft man will, aus dem Speicher herausgeholt werden können, vorausgesetzt, man kennt ihren Namen.

Nichts ist besser als Ausprobieren. Geben Sie direkt ein:

```
A=25:B=50:C$="TEXT"    (<RETURN>-Taste drücken)
```

Damit haben wir einer Variablen mit Namen A den Wert 25 zugewiesen, einer anderen mit Namen B den Wert 50 und einer dritten, nämlich C$ das Wort »TEXT«. Mit:

```
PRINT A:PRINT B  (<RETURN>)
```

können wir die A und B zugewiesenen Werte jederzeit aus dem Speicher auf den Bildschirm bringen.

Oben haben wir noch eine 3. Variable eingegeben, nämlich C$. Das Dollarzeichen $ kennzeichnet sie als eine »String-Variable«, das heißt, ihr Wert ist keine Zahl, sondern eine Buchstabenfolge, auch Zeichenkette oder String genannt. Ein String muß immer zwischen Anführungszeichen stehen. Auch kann er mittels:

```
PRINT C$  (<RETURN>)
```

aus dem Speicher geholt werden. Soweit die kurze Wiederholung aus dem Anfangskurs, außerdem möchte ich als Basis für die dann folgende Erweiterung des Variablen-Konzeptes die entsprechende Zusammenfassung wiederholen.

Zusammenfassung

1. Wir haben von zwei Variablen-Typen gesprochen:
- den »numerischen« Variablen weisen wir ausschließlich Zahlenwerte zu
- den »String-Variablen«, gekennzeichnet durch das $-Zeichen am Ende des Variablennamens, weisen wir ausschließlich Zeichen, Buchstaben oder ganze Folgen von ihnen zu. Diese Folgen heißen »Zeichenketten« oder »Strings«.

Zahlen sind auch hier zugelassen, nur werden sie wie Zeichen (und nicht wie Zahlenwerte) behandelt.

2. Die Zuordnung von Zahlen oder Strings zum falschen Variablentyp führt immer zu einer Fehlermeldung und oft auch zum Abbruch des Programms.
3. Der Name einer Variablen darf theoretisch aus soviel Zeichen bestehen, wie in eine Programmzeile hineinpassen. Das ist aber wenig sinnvoll, da der Computer nur die ersten beiden Zeichen als Name der Variablen registriert (PROFIT und PRODUKT ist für ihn somit ein und dieselbe Variable).
4. Im Namen einer Variablen darf kein Basic-Befehl enthalten sein: Namen wie OTTO oder SPRINT werden vom Computer mit einer Fehlermeldung zurückgewiesen.

Der Punkt 4 oben ist neu für uns; ich habe im Anfangskurs nichts davon erwähnt.

Aber diese Regel ist wichtig. Probieren Sie es ruhig aus:

```
SPRINT=25     (<RETURN>)
OTTO=2        (<RETURN>)
```

führen unweigerlich zur Fehlermeldung SYNTAX ERROR. Warum? Nun, im ersten Wort ist der Befehl PRINT enthalten, das S davor wird als falsche Schreibweise dieses Befehls angesehen und in OTTO ist der Befehl TO enthalten.

Das ist übrigens ein häufiger Fehler, denn wer hat schon alle Basic-Befehle im Kopf. Es ist daher ratsam, auf »schöne« Variablennamen zu verzichten und Variable lieber A3 oder ZZ zu nennen.

Gerade die Kombination eines Buchstabens mit einer Zahl ist sehr vielseitig. Voraussetzung ist, wie Sie sicher wissen, daß der Name mit einem Buchstaben anfängt.

4. Variablen in Feldern (Arrays)

Nun lade ich Sie zu einem Experiment ein.

Wir stellen uns vor, wir benötigten in einem Programm eine Tabelle mit Zahlen, zum Beispiel die Anzahl der Bewohner, die in 10 einstöckigen Häusern einer langen Straße wohnen. Die Adresse der Häuser unterscheidet sich nur durch die Haus-

nummer, denn der Straßenname bleibt ja gleich. Statt des langen Straßennamens nehmen wir nur einen Buchstaben, zum Beispiel T. Zur Kennzeichnung der 10 verschiedenen Adressen hängen wir die Ziffern 0 bis 9 hinten an. Die Zahl 10 können wir nicht nehmen, weil nach nach Punkt 3 der obigen Zusammenfassung die zweistelligen Variablennamen T1 und T10 identisch sind.

In unserem Beispiel wollen wir den einzelnen Häusern Bewohnerzahlen zuweisen, die jeweils um 2 größer sind als die Hausnummer.

```
10 T0=2
20 T1=3
30 T2=4
40 T3=5
 .
 .
 .
100 T9=11
110 PRINT T0;T1;T2;T3;T4;T5;T6;T7;T8;T9
```

Das ist natürlich eine sehr umständliche Arbeit, und langweilig ist diese dauernde Wiederholung obendrein. Also, wie geht das einfacher und eleganter? Natürlich mit einer FOR-TO-NEXT-Schleife. Aber aufgepaßt: bei ihrer Verwendung müssen wir die Hausnummer N in Klammern dem Straßennamen T anhängen. Anders akzeptiert das der C 16 in Basic nicht. Als Programm sieht das so aus:

```
10 FOR N=0 TO 9
20 T(N)=N+2
30 PRINT T(N)
40 NEXT N
```

Ihnen ist natürlich klar, daß Straßenname T plus Hausnummer (N) eine Variable ist, der wir Werte (Bewohner) zuweisen.

Aber eins sollte Sie stutzig machen. Wie unterscheiden sich diese speziellen Variablen voneinander? Sind nicht T(3) und T(8) identisch, wenn nach der oben schon zitierten Regel nur die ersten beiden Zeichen des Namens – hier T(– hergenommen werden?

Nun, die Klammer macht den Unterschied. Durch sie wird ein neuer Typ einer Variablen, eine sogenannte *Feld-Variable* definiert, und Feld-Variablen unterscheiden sich durch die Zahl innerhalb der Klammer. Diese Zahl heißt *Index* (Mehrzahl: Indizes). Warum sie ausgerechnet Feld-Variable heißt, erkläre ich gleich.

Vorher, bei der Schreibweise T0,T1 etc. waren wir auf 10 Variablen beschränkt, weil T1 und T10 dieselbe Variable war.

Jetzt – mit Indizes als Unterscheidungsmerkmal – gilt diese Beschränkung nicht. Deshalb wollen wir die Anzahl der Indizes im Prögrämmchen noch erhöhen. Bei einem N von 0 bis 10 geht es ohne Probleme. Aber ab 11 ist schon wieder Feierabend. Wir erhalten die Fehlermeldung »BAD SUBSCRIPT ERROR IN 20«. Um Ihnen das zu erklären, muß ich genauer beschreiben, was passiert, wenn wir eine Variable mit einem Index in Klammern verwenden.

Wenn wir eine Variable so schreiben:

T(1)=25

dann nimmt der Computer an, daß wir in einer Tabelle noch mehrere derartige Variable T( ) verwenden wollen – klar, sonst würden wir uns ja die Mühe mit der Klammer nicht machen.

Um uns die Sache zu erleichtern, reserviert der Computer unter dem Variablennamen T in seinem Speicher von vornherein 11 Plätze, für T(0) bis T(10). Diesen reservierten Bereich können Sie salopp *Tabelle* nennen; offiziell heißt er *Feld* oder auf englisch *Array*. Die alte Regel für den Namen der Variablen gilt jetzt auch nicht mehr, denn zum Unterscheiden der einzelnen Feld-Variablen desselben Anfangsbuchstabens bedient der Computer sich der in Klammer stehenden Index-Zahl.

Die Beschränkung auf eine Feld-(Array-)Größe von 11 Plätzen wäre natürlich sehr lästig, wenn sie nicht umgangen werden könnte. Wenn wir nämlich mehr Platz brauchen, können wir dem Computer mit dem Basic-Befehl

DIM

unsere Reservierungswünsche mitteilen. DIM ist eine Abkürzung, die aus dem Wort »Dimension« abgeleitet ist. Als Beispiel möge die folgende Programmzeile dienen:

```
5 DIM T(25)
```

Sie reserviert im obigen Programm für die Variable T( ) im Speicher ein Feld von 26 Plätzen, von T(0) bis T(25).

Die Größe eines Feldes ist nur durch den vorhandenen Speicherplatz begrenzt. Wenn Sie die Zahl zu groß wählen, verweigert der Computer die Reservierung – natürlich erst nach dem Befehl RUN – mit OUT OF MEMORY, was soviel heißt wie KEINEN PLATZ MEHR IM SPEICHER.

DIM-Befehl

- dieser Befehl wird in der folgenden Schreibweise verwendet:
  DIM Variablenname (Index)
- er reserviert einen Speicherbereich für eine durch den Index festgelegte Anzahl von Variablen desselben Namens, die sich nur durch ihren jeweiligen Index unterscheiden. Dieser Speicherbereich wird Feld oder Array genannt.
- mit DIM können sowohl Felder für numerische, als auch für String-Variable reserviert werden. Ein Feld kann immer nur einen einzigen Variablentyp enthalten.
- für den Namen und für die Kennzeichnung des Typs der Feld-Variablen (vor der Klammer) gelten dieselben Regeln wie für alle anderen Variablen.

Ein Feld kann also auch aus String-Variablen bestehen:

```
200 DIM A$(25)
210 A$(0)="FEUER"
220 A$(1)="ZANGEN"
230 A$(2)="BOWLE"
 .    .
 .    .
 .    .
300 FOR N=0 TO 2
310 PRINT A$(N);
320 NEXT N
```

Dieses kleine Programm reserviert ein Feld von 25 Strings und weist den ersten drei Strings davon je ein Wort zu. Mit RUN 200 gestartet, druckt es das Wort FEUERZANGENBOWLE aus, indem in der Zeile 310 nacheinander für die Werte von N=0 bis N=2 die gespeicherten Strings A$(0) bis A$(2) durch das Semikolon nach dem PRINT-Befehl aneinandergeklebt werden. Das Eintragen der einzelnen Werte in die Tabelle, das ich in den Zeilen 210 bis 230 vorgenommen und danach nur noch angedeutet habe, geht viel eleganter mit den Befehlen READ – DATA, die im Anfangskurs auf Seite 65 beschrieben sind.

Um das zu zeigen, schlage ich Ihnen eine kleine Anwendung – im Computerdeutsch heißt so ein Programm »Utility« – vor, und zwar eine Nachschlagliste von Geburtstagen Ihrer Freunde und Verwandten. Natürlich ist das kein Utility-Programm, wegen dem ich mir einen Computer kaufen würde, aber für unsere Zwecke hier ist es ganz passend.

Ich lege das Beispiel vorerst einmal auf 5 Namen (N$) und dazugehörige Geburtstagsdaten (D$) aus. Wir brauchen also 2 Felder mit je 5 Plätzen, dazu zwei DATA-Zeilen mit den Eintragungen. Verwenden Sie bitte meine wirr erscheinenden Zeilennummern; am Ende passen sie schon zusammen.

```
20 DIM N$(4)
500 DATA MAX,MORITZ,MARIA,HANS,LUISE
120 DIM D$(4)
600 DATA 12.6.52,3.4.60,21.1.40,19.9.56,11.11.70
```

Denken Sie bitte daran: DIM N$(4) legt 5 Plätze fest, nämlich 0 bis 4.

Denken Sie ebenfalls daran, daß Eintragungen in DATA-Zeilen durch ein Komma voneinander getrennt sein müssen.

So, jetzt brauchen wir für beide Felder eine Schleife, mit der die Eintragungen der DATA-Zeilen mit READ in das jeweilige Feld gelesen werden. Für das Namensfeld gilt:

```
30 FOR I=0 TO 4
40 READ N$(I)
50 NEXT I
```

Dasselbe machen wir für das zweite Feld der Geburtstage:

```
130 FOR I=0 TO 4
140 READ D$(I)
150 NEXT I
```

Jetzt brauchen wir noch einen Programmteil, der nach Eingabe eines Namens das dazugehörige Geburtstagsdatum heraussucht und ausdruckt. Für die »Dazugehörigkeit« benützen wir die Indizes der Feld-Variablen. Das bedeutet nichts anderes, als daß zum zweiten Namen N$(2) das zweite Datum D$(2) gehört.

Doch zuerst stellen wir per INPUT die Frage F$ nach dem Namen, dessen Geburtstag wir wissen wollen:

```
200 INPUT "NAME";F$
```

Dann vergleichen wir in einer weiteren Schleife die gespeicherten Namen N$(I) mit dem gefragten Namen F$. Wenn bei einem bestimmten Wert I der Vergleichsschleife die beiden Namen gleich sind, wird der Geburtstag mit demselben Index I ausgedruckt. Das Programm kann dann eine weitere Eingabe eines Namens verlangen (GOTO 200).

```
210 FOR I=0 TO 4
220 IF N$(I)=F$ THEN PRINT D$(I):GOTO 200
230 NEXT I
```

Um den Vergleichs- und Entscheidungsvorgang genau zu verstehen, brauchen Sie lediglich die Schleifen durchzuspielen, indem Sie die Werte von I gedanklich hochzählen und im Programm nachschauen, was passiert. Wir können aber zum Verständnis auch eine »Lehrzeile« (mit h!) einfügen, die uns den jeweiligen Stand des Vergleiches anzeigt:

```
215 PRINT I;F$,N$(I);D$(I)
```

Nach RUN und nach Eingabe eines der fünf Namen sehen wir am Bildschirm eine Reihe mit steigenden Werten von I, daneben (Semikolon!) den eingegebenen Namen, dann in einem größeren Abstand (Komma!) die Namen und Daten, und zwar solange, bis F$ und N$ gleich sind.

Nehmen Sie jetzt Zeile 215 wieder heraus, aber fügen Sie bitte eine Zeile 240 dazu, die wir für ein bedienungsfreundliches Programm brauchen. Wir müssen nämlich Vorkehrung für den Fall treffen, daß ein Name eingegeben wird, der nicht in der Liste steht. Auch Tippfehler fallen in diese Kategorie. Wenn kein positiver Vergleich zwischen F$ und N$ innerhalb der Schleife auftritt, macht das Programm nach der Schleife weiter, mit der neuen Zeile 240, die für sich selbst spricht:

```
215
240 PRINT"NAME IST NICHT IN DER LISTE"
```

Das ganze Programm sieht so aus:

```
20 DIM N$(4)
30 FOR I=0 TO 4
40 READ N$(I)
50 NEXT I
120 DIM D$(4)
130 FOR I=0 TO 4
140 READ D$(I)
150 NEXT I
200 INPUT "NAME";F$
210 FOR I=0 TO 4
220 IF N$(I)=F$ THEN PRINT D$(I):GOTO 200
230 NEXT I
240 PRINT"NAME IST NICHT IN DER LISTE"
500 DATA MAX,MORITZ,MARIA,HANS,LUISE
600 DATA 12.6.52,3.4.60,21.1.40,19.9.56,11.11.70
```

Der DIM-Befehl und der READ-Befehl haben beide eine zusätzliche Eigenschaft, die uns eine wesentliche Verbesserung dieses Programms erlaubt:

Man darf hinter einem einzigen DIM-Befehl mehrere Felder dimensionieren. Sie müssen lediglich durch ein Komma getrennt sein. Genauso darf man mit einem einzigen READ-Befehl mehrere Werte-Gruppen auslesen. Hier gilt dieselbe Kommaregel wie bei DIM.

Damit können wir die Zeilen 20 bis 150 stark verkürzen:

```
20 DIM N$(4),D$(4)
30 FOR I=0 TO 4
40 READ N$(I),D$(I)
50 NEXT I
```

Die Zeile 20 ist einfach nur eine Zusammenziehung der beiden alten Zeilen 20 und 120. Die Funktion des DIM-Befehls bleibt dabei erhalten – er dimensioniert halt gleich beide Felder.

Beim »doppelten« READ-Befehl in Zeile 40 hat sich in der Arbeitsweise, verglichen mit den alten Zeilen 40 und 140, etwas geändert. Schuld daran ist aber die FOR-NEXT-Schleife der Zeile 30.

Bei jedem Wert von I holt sich der READ-Befehl zwei hintereinanderliegende Werte aus den DATA-Zeilen und weist sie den beiden Variablen N$ und D$ zu. Wir müssen diesem Vorgang dadurch Rechnung tragen, daß wir den Inhalt der DATA-Zeilen umorganisieren. Es müssen jetzt immer je ein Name und das dazugehörige Geburtsdatum hintereinander kommen – was eigentlich viel leichter einzutippen ist.

Sie sehen, gute Programmierung ist fast immer auch klarer und logischer.

```
500 DATA MAX,31.3.55,MORITZ,12.4.45,HANS,6.2.60
600 DATA MARIA,14.7.63,LUISE,8.9.60
```

Diese Anordnung bietet noch einen weiteren Vorteil. Das Programm, oder besser gesagt die Geburtstagskartei, kann ganz leicht erweitert werden. Einen neuen Namen samt Datum brauchen Sie nur durch Komma getrennt hinter die letzte Eintragung der DATA-Zeilen zu schreiben.

Doch halt! Noch etwas fehlt: Die Zahl der Namen, die sich durch eine neue Eintragung natürlich erhöht, kommt ja im Programm auch vor, und zwar in unserem Fall als Ziffer 4 in den Zeilen 20, 30 und 210. Um uns diese Zusatzarbeit ebenfalls zu erleichtern, geben wir dieser Zahl einen Variablennamen Z, definieren dieses Z ganz am Anfang und brauchen so bei jeder neuen Namenseintragung nur dieses Z um 1 erhöhen.

Diese Zeilen sehen jetzt so aus:

```
10 Z=4
20 DIM N$(Z),D$(Z)
30 FOR I=0 TO Z
210 FOR I=0 TO Z
```

Zusammenfassung DIM-Befehl

1. Hinter einem DIM-Befehl können mehrere Felder dimensioniert werden. Sie müssen lediglich durch ein Komma getrennt sein.
2. Mit einem READ-Befehl können mehrere DATA-Werte gelesen werden. Auch diese müssen durch ein Komma voneinander getrennt werden.
3. Es empfiehlt sich, einen Zahlenwert oder einen String, der mehrmals in einem Programm vorkommt, ganz am Anfang des Programms als numerische- beziehungsweise als String-Variable zu definieren und im Programm dann nur einmal dieser Variablen den Wert zuzuweisen.

Wir haben jetzt ein komplettes kleines Programm, dessen zusammengewürfelte Zeilennummern ihm allerdings ein unfertiges Aussehen verleihen.

Hier hilft uns ein Basic-Befehl, der Zeilennummern »umnumeriert«. Er heißt *RENUMBER*.

Wenn wir diesen Befehl ohne Zusätze verwenden:

```
RENUMBER (<RETURN> nicht vergessen)
```

dann erhält das kleine Programm neue Zeilennummern, die mit 10 anfangen und in Zehnerschritten weitergehen. Sehr wichtig dabei ist, daß RENUMBER die Zeilennummer hinter dem GOTO-Befehl automatisch auf den richtigen neuen Wert umändert. Mit LIST können Sie das Resultat überprüfen.

Das ist aber noch nicht alles. Eine Zahl hinter dem RENUMBER-Befehl läßt die Zeilennumerierung eines Programms ab dieser Zahl beginnen.

```
RENUMBER 1000
LIST
```

beweist es Ihnen.

Zu guter Letzt kann auch noch die Schrittweite, also der Zahlenabstand zwischen den Programmzeilen, ausgewählt werden durch Angabe einer zweiten Zahl:

```
RENUMBER 10,100
```

gibt uns ein Programm, desssen erste Zeile mit 10 beginnt, alle anderen Zeilen dagegen mit einem Abstand von 100 weiterzählen.

RENUMBER-Befehl

- Dieser Befehl numeriert ein im Arbeitsspeicher des Computers gespeichertes Programm neu durch.
- »RENUMBER« numeriert das Programm ab Zeile 10 in Zehnerschritten durch.
- »RENUMBER Z« numeriert das Programm ab Zeile Z in Zehnerschritten durch.
- »RENUMBER Z,S« numeriert das Programm ab Zeile Z mit der Schrittweite S durch.
- Dabei werden alle Zeilennummern am Beginn einer Zeile und hinter den Basic-Befehlen GOTO, GOSUB, THEN und ELSE entsprechend angeglichen.

Für unser Programm schlage ich vor, ab Zeile 10 zu beginnen und in Zehnerschritten weiterzumachen.

5. Grafische Verschönerungsmittel

Der C16 besitzt erstaunliche grafische Fähigkeiten.

Als erstes wollen wir das Resultat der Datumssuche, nämlich das ganz unscheinbar hingedruckte Geburtsdatum, hervorheben. Dazu gibt uns der C16 drei Hilfsmittel:
- farbige Schrift
- invertierte Schrift
- blinkende Schrift

Im Direkt-Modus, das heißt ohne Zeilennummern direkt eingetippt, kann die Farbe der Zeichen und Buchstaben durch gleichzeitiges Drücken der <CTRL>-Taste beziehungsweise <Commodore>-Taste und einer der acht Farbtasten eingestellt werden.

Beide Vorgänge sind im Commodore-Handbuch auf Seite 49 bis 51 gut beschrieben.

Auf Seite 23 sind zwei weitere Tasten genannt, <FLASH-ON> und <FLASH-OFF>, die gleichzeitig mit <CTRL> gedrückt einen Blinkeffekt hervorrufen.

Wie können wir dasselbe im Programm-Modus machen?

Im Anfangskurs habe ich ab Seite 58 unter der Überschrift »Programmierte Steuertasten« diese Vorgehensweise bereits beschrieben. Sie verwendet die sogenannten ASCII-Codewerte der Steuer- und Farbtasten, die mit dem PRINT-Befehl die Funktionen dieser Tasten ausführen.

Die ASCII-Codewerte stehen dabei in Klammern hinter dem Basic-Befehl CHR$.

```
PRINT CHR$(18)"ABC"
```

zum Beispiel druckt die drei Buchstaben in invertierter Darstellung.

Wichtig ist natürlich die Kenntnis aller entsprechenden ASCII-Werte. Im Bedienungshandbuch von Commodore ist ab Seite 215 eine Codetabelle enthalten, nur ist sie leider nicht ganz vollständig. Ich habe daher in Tabelle 1 die ASCII-Codewerte und die invertierten Zeichen aller Steuertasten zusammengestellt. Die invertierten Zeichen treten bekanntlich dann auf, wenn im Anführungszeichen-Modus (nach einem Anführungszeichen) eine Taste gedrückt wird.

Als erstes wollen wir das Ausdrucken des Geburtstages in Zeile 80 (vor dem RENUMBER hatte sie die Nummer 220) durch den Zusatz CHR$(18) invertieren:

```
80 IF N$(I)=F$ THEN PRINT CHR$(18) D$(I):GOTO 60
```

Der Zusatz des Codewertes 130 läßt den Datumsausdruck blinken:

```
80 IF N$(I)=F$ THEN PRINT CHR$(18) CHR$(130) D$(I)
:GOTO 60
```

Wenn wir jetzt noch weitere CHR$-Befehle hinzufügen, dann wird langsam die Zeile zu klein – ein Grund für mich, die platzsparende Anführungszeichen-Methode vorzuführen.

Tippen Sie bitte in Zeile 80 nach dem Befehl PRINT ein Anführungszeichen und danach – gleichzeitig mit der <CTRL>-Taste – die <RVS-ON>-Taste, die <FLASH-ON>-TASTE und die Taste für die Farbe <BLAU> (<BLUE>). Schließen Sie bitte noch vor dem D$(I) das Ganze mit einem Schlußzeichen (zweites Anführungszeichen) ab. Alle CHR$-Befehle von vorher müssen Sie löschen.

Auf dem Bildschirm steht jetzt:

```
80 IF N$(I)=F$ THEN PRINT "{rvs on}{flash-
on}{blue}" D$(I):GOTO 60
```

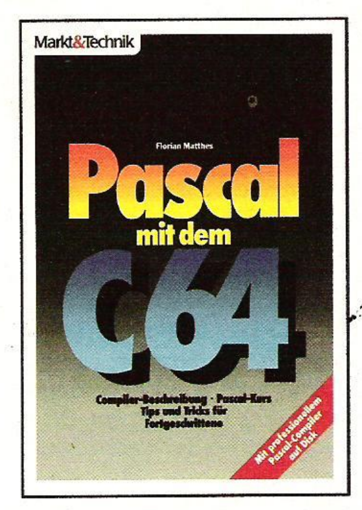
FUNKTION	TASTE(N)	ZEICHEN	ASCII
schwarz	CTRL-1		144
weiß	CTRL-2		5
rot	CTRL-3		28
lila	CTRL-4		159
purpur	CTRL-5		156
grün	CTRL-6		30
blau	CTRL-7		31
gelb	CTRL-8		158
orange	C=-1		129
braun	C=-2		149
gelbgrün	C=-3		150
rosa	C=-4		151
blaugrün	C=-5		152
hellblau	C=-6		153
dunkelblau	C=-7		154
hellgrün	C=-8		155
Revers ein	CTRL-9		18
Revers aus	CTRL-0		146
Cursor nach oben	CRSR↑		145
Cursor ab	CRSR↓		17
Cursor links	CRSR←		157
Cursor rechts	CRSR→		29
Schirm löschen	SHIFT-CLEAR/HOME		147
Cursor Home	CLEAR/HOME		19
Insert	SHIFT-INST/DEL		148
Delete	INST/DEL		20
Return	RETURN		13
Shift-Return	SHIFT-RETURN		141
Escape	ESC		27
Blinken ein	CTRL-,		130
Blinken aus	CTRL-.		132
Klein/Großbuchstaben	SHIFT-C=		14
Großbuchstaben/Zeichen	SHIFT-C=		142
Funktionstaste f1	F1		133
f2	F2		137
f3	F3		134
f4	SHIFT-F1		138
f5	SHIFT-F2		135
f6	SHIFT-F3		139
f7	SHIFT-HELP		136
f8	HELP		140

Tabelle 1. Tasten, reverse Zeichen und ASCII-Codes aller Steuertasten des C64

Bücher zum Commodore 64

H. Haberl
Mini-CAD mit Hi-Eddi plus auf dem C64/C128
1985, 234 Seiten inkl. Diskette
Zusammen mit diesem Buch erhalten Sie auf Diskette ein CAD-Programm, welches das komfortable Erstellen von technischen Zeichnungen, Plänen oder Diagrammen ebenso erlaubt wie das Malen von Bildern in Farbe, Schildern Briefköpfen und sogar das Erstellen von kleinen Trickfilmen oder von Schaufensterwerbung.
Bei dem Programm handelt es sich um eine stark verbesserte Version des Zeichenprogramms »Hi-Eddi« aus der Zeitschrift 64'er. Das Buch beschreibt ausführlich und auch für Laien verständlich die Anwendungsmöglichkeiten an einer Reihe von Beispielen wie das Erstellen von Schaltplänen, Platinenlayouts und Struktogrammen. Dazu bietet es durch eine ausführliche Dokumentation dem maschinenspracheerfahrenen Programmierer die Möglichkeit, Grafikroutinen in eigene Programme zu übernehmen. Hardwareanforderungen: Floppy 1525/MPS801/MPS803, Joystick.
Best.-Nr. MT 90136
ISBN 3-89090-136-0
DM 48,–/sFr. 44,20/öS 374,40



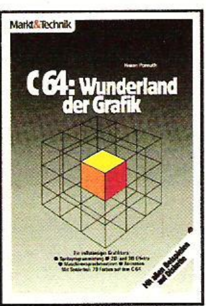


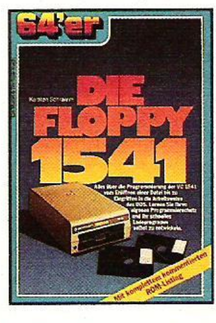
F. Matthes
Pascal mit dem C64
Mai 1986, 215 Seiten inkl. Diskette
Buch und Compiler ermöglichen jedem Besitzer eines C64 den Einstieg in die moderne Programmiersprache Pascal.
Dem Anfänger wird ein Einführungskurs in Pascal geboten, wobei viele überschaubare Beispiele aus der Praxis und Übungsaufgaben zum aktiven Lernen mit dem C64 auffordern. Beim Programmieren wird er durch eine ausführliche Bedienungsanleitung unterstützt.
Für den Pascal-Profi gibt es neben nützlichen Beispielprogrammen ein spezielles Kapitel mit Tips und Tricks. Der Compiler akzeptiert den gesamten Sprachumfang mit einigen Erweiterungen. Er bildet mit dem sehr komfortablen Full-Screen-Editor eine schnelle Einheit, so daß der Programmentwicklungsaufwand minimal ist. Übersetzte Programme laufen ohne weitere Hilfsprogramme auf jedem C64, nutzen den gesamten Programmspeicher des C64 und sind 3–4mal schneller als vergleichbare Programme in BASIC.
Best.-Nr. MT 90222
ISBN 3-89090-222-7
DM 52,–/sFr. 47,80/öS 405,60

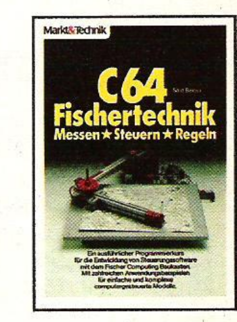












W. Kassera/F. Kassera
C64 Programmieren in Maschinensprache
1985, 327 Seiten inkl. Disk
Die Autoren zeigen in diesem Buch, daß jeder die Grenzen des eingebauten BASIC des C64 sprengen kann. Der Aufschwung im Programmieren stellt sich ein, wenn Sie effektiv die betriebssystem-internen ROM-Routinen nutzen können. Dazu aber müssen Sie diese Routinen kennen, müssen über ihre Funktionsweise und ihr Zusammenspiel informiert sein. Und Sie müssen die Maschinensprache Ihres C64 beherrschen. Beides ermöglicht Ihnen dieses Buch: die Programmierung bewegter Bildschirmobjekte, die Erweiterung der Interrupt-Routine des Systems, die Arithmetik-Routinen im ROM.
Best.-Nr. MT 830
ISBN 3-89090-168-9
DM 52,–/sFr. 47,80/öS 405,60

W.-J. Becker/M. Folprecht
Programmieren unter CP/M mit dem C 64
1985, 290 Seiten
DAS Buch für den C64- und CP/M-Freak! Selbstverständlich wird alles vermittelt, was für die Arbeit mit CP/M nötig ist: die Kommandosprache und deren residente und transiente Kommandos, aber auch die verbreitetsten Programmiersprachen (BASIC-80 und MBASIC, Nevada-FORTRAN und FORTRAN-80 sowie Turbo-Pascal). Systemprogrammierer finden ein ausführlich kommentiertes BIOS-Listing und Listings der wichtigsten System-Dienstprogramme sowie Bestückungsplan, Blockschaltbild und Schaltbild des CP/M-Moduls.
Best.-Nr. MT 751
ISBN 3-89090-091-7
DM 52,–/sFr. 47,80/öS 405,60

H. L. Schneider/W. Eberl
Das C64 Profihandbuch
1985, 413 Seiten
Der erste Teil des Buches widmet sich allgemeinen Algorithmen zu unterschiedlichen Problemen (Sortieren, Menütechniken, Datumsverarbeitung), die zum Wissensstand jedes Profis gehören müssen. Es folgen nützliche Utilities in BASIC und Maschinensprache, die auch zu Erweiterungen des eingebauten BASIC genutzt werden können. Ein Kapitel über die Schnittstellen zur Außenwelt wird gefolgt von systemnahen Tips und Tricks mit PEEK, POKE und SYS. Der technische Teil mit Tabellen zu Hard- und Software (BS-Routinen und Einsprungpunkte, Bausteine, Zeichensatz-Generator, Fehler von Rechner und Floppy etc.) runden das Buch ab.
Best.-Nr. MT 749
ISBN 3-89090-110-7
DM 52,–/sFr. 47,80/öS 405,60

H. Ponnath
C64: Wunderland der Grafik
1985, 232 Seiten inkl. Disk
Wenn sie nicht gerade von der allereinfachsten Art sein soll, dann setzt Grafikprogrammierung auf dem C64 einige Kenntnisse des Systems voraus: man bewegt sich meist auf der Ebene der Maschinenprogrammierung. Aber keine Angst! Der Autor legt beim Leser ein solides Fundament an Wissen und er tut dies auch noch auf so unterhaltsame Art, daß Sie bestens gerüstet sind, um so interessante Aufgaben wie die Programmierung hochauflösender zwei- und dreidimensionaler Grafiken anzugehen. Mit Sprites zu jonglieren ist für Sie bald kein Problem mehr.
Best.-Nr. MT 90363
ISBN 3-89090-363-0
DM 49,–/sFr. 45,10/öS 382,20

K. Schramm
Die Floppy 1541
April 1985, 434 Seiten
Egal, ob Sie als Floppy-Einsteiger nur wissen wollen, wie man mit der 1541 Daten speichern kann oder ein Perfektionist sind, der jedes Detail seines Diskettenlaufwerks beherrschen will: In diesem Buch werden Sie alle Informationen über Ihre Floppy finden; für den Anfänger beginnend bei der Handhabung der Kanäle und der verschiedenen Filetypen bis hin zum gut kommentierten DOS-Listing der 1541 für Assemblerprofis.
Best.-Nr. MT 90098
ISBN 3-89090-098-4
DM 49,–/sFr. 45,10/öS 382,20

S. Baloui
C64 Fischertechnik Messen, Steuern, Regeln
Februar 1986, 174 Seiten
Dieses Buch bietet einen ausführlichen Programmierkurs für die Entwicklung von Steuerungssoftware mit dem Fischer- Computing-Baukasten. Es beginnt mit einer grundsätzlichen Darstellung der verschiedenen Elemente »Lampen, Motoren, Elektromagnete, Potentiometer«, den jeweiligen Einsatzmöglichkeiten und der Verkabelung. Danach folgt die Darstellung des zugehörigen Interfaces.
Best.-Nr. MT 90194
ISBN 3-89090-194-8
DM 29,90/sFr. 27,60/öS 233,20



Bestellungen im Ausland bitte an den Buchhandel oder an untenstehende Adressen.
Schweiz: Markt & Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, ☎ 042/41 56 56
Österreich: Ueberreuter Media Handels- und Verlagsges. mbH, Alser Straße 24, 1091 Wien, ☎ 0222/48 15 38-0

Irrtümer und Änderungen vorbehalten.



Unternehmensbereich Buchverlag
Hans-Pinsel-Straße 2, 8013 Haar bei München



Beachten Sie bitte die Schreibweise für die »Steuerzeichen«. {rvs on} bedeutet, daß hier das Steuerzeichen für »Invertierte Schrift Ein« steht; ein revers geschriebenes »R«.

Nach RUN erscheint der Datumsausdruck invertiert, blinkend und in blauer Farbe.

PRINT "{flash-on}" erreicht dasselbe wie PRINT CHR$(130)

PRINT "{blue}" erreicht dasselbe wie PRINT CHR$(31)

Die Gründe, warum die invertierten Steuerzeichen in Zeitschriften und gedruckten Listings selten vorkommen, liegen einmal in dem nicht ganz einfachen Druck dieser Zeichen, zum andern in ihrer schlechten Lesbarkeit.

Anmerkung: Beim C16 sind nach dem Einschalten des Computers die acht Funktionstasten mit festen Funktionen belegt. Die in der Tabelle genannten ASCII-Codewerte stehen deswegen erst nach einer Löschung dieser Funktionen zur Verfügung. Dieser Vorgang wird bei der Behandlung des Basic-Befehls KEY erklärt.

Ich ziehe normalerweise den CHR$-Befehl vor, es sei denn, ich gerate in Platznot, so wie oben.

Eine letzte Verfeinerung des Programms sei mir gestattet. Im Gegensatz zu <REVERS-ON> und <FLASH-ON>, die beide nur innerhalb einer einzigen Programmzeile wirken, bleibt eine Farbumschaltung für alle folgenden Programmteile erhalten. Wir müssen also nach dem PRINTen von D$(I) wieder auf die ursprüngliche Zeichenfarbe schwarz (black) zurückschalten:

```
80 IF N$(I)=F$ THEN PRINT"{rvs on}{flash on}{blue}
"D$(I) "{black}":GOTO 60
```

In Listing 1 ist das ganze Programm, so wie es jetzt steht, zusammengefaßt abgedruckt:

```
10 Z=4
20 DIM N$(Z),D$(Z)
30 FOR I=0 TO Z
40 READ B$(I),D$(I)
50 NEXT I
60 INPUT "NAME";F$
70 FOR I=0 TO Z
80 IF N$(I)=F$ THEN PRINT"{rvs on}{flash on}" D$(I)
   "{black}" :GOTO 60
90 NEXT I
100 PRINT"NAME NICHT IN DER LISTE"
110 DATA MAX,31.3.55,MORITZ,12.4.45,HANS,6.2.60
120 DATA MARIA,14.7.63,LUISE,8.9.60
```

6. Zweidimensonale Felder (Arrays)

Hinter dieser Überschrift verbirgt sich eine Erweiterung des DIM-Befehls:

- eindimensional ist eine Linie; sie hat nur in einer Richtung eine Ausdehnung
- zweidimensional ist eine Fläche; sie hat eine Länge und eine Breite.

Diese Betrachtungsweise kann auch auf Variablenfelder angewendet werden.

Ich habe bei der ersten Erklärung des Begriffs »Feld« und »Feldvariable« den Vergleich mit einstöckigen Häusern in einer Straße verwendet. Unser damaliges Beispiel könnte man so aufmalen:

Bewohner	2	3	4	5	6
Adresse	T(0)	T(1)	T(2)	T(3)	T(4)

Ein zweidimensionales Feld ist eine Tabelle, in der sowohl waagrecht wie senkrecht Eintragungen möglich sind:

In unserem Beispiel sind das Häuser, die mehrere Stockwerke und pro Stockwerk verschiedene Bewohnerzahlen haben. Der Einfachheit halber gebe ich allen Häusern dieselbe Zahl von Stockwerken.

Stockwerk						
	3	T(0,3)	T(1,3)	T(2,3)	T(3,3)	T(4,3)
	2	T(0,2)	T(1,2)	T(2,2)	T(3,2)	T(4,2)
	1	T(0,1)	T(1,1)	T(2,1)	T(3,1)	T(4,1)
	0	T(0,0)	T(1,0)	T(2,0)	T(3,0)	T(4,0)
Hausnummer		0	1	2	3	4

Bei diesen Reihenhäusern habe ich nicht mehr die Zahl der Bewohner in die Häuser beziehungsweise Stockwerke geschrieben, sondern die »Adressen«, die wieder unsere Feld-Variablen sind. Nur haben sie diesmal zwei Indizes, einen für das Stockwerk (Zeile des Feldes) und einen für die Hausnummer (Spalte des Feldes).

Wenn wir jetzt sagen wollen, daß im 3. Stock des 1. Hauses 7 Leute wohnen, legen wir das fest mit:

T(1,3)=7

Bei der Definition einer zweidimensionalen Variablen T(A,B) reserviert auch diesmal wieder der Computer ein Feld von 11 Variablen, allerdings pro Index, das heißt insgesamt 11 x 11=121 Plätze.

Ein kleines Programm beweist diese Behauptung:

```
10 FOR I=0 TO 10
20   FOR K=0 TO 10
30     T(I,K)=I+K
40     PRINT T(I,K);
50   NEXT K
60 NEXT I
```

Ich habe es extra so geschrieben, daß Sie die beiden Schleifen besser sehen können. Pro Durchlauf der äußeren I-Schleife läuft die innere K-Schleife 11mal durch. Dementsprechend sieht das Resultat der Zeile 40 aus.

Wenn Sie jetzt die obere Grenze von I auf 11 erhöhen, merkt das Programm erst nach dem 121. Durchlauf, daß zuwenig Platz reserviert worden ist. Lassen Sie dagegen das I auf maximal 10, erhöhen aber das K auf 11, bleibt das Programm schon nach dem ersten Durchlauf der inneren Schleife stehen.

Das war eine kleine Erinnerung an die Wirkungsweise geschachtelter Schleifen.

Wenn wir ein größeres Feld benötigen, müssen wir diesen Bedarf wieder mit dem DIM-Befehl eingeben:

DIM T(25,34)

Zusammenfassung DIM-Befehl

1. Felder können auch mehrere Dimensionen haben. Eine zweidimensionale Feld-Variable hat 2 Indizes in der Klammer, die durch ein Komma getrennt sein müssen.
2. Für eine mehrdimensionale Feld-Variable werden pro Index 11 Plätze im Speicher reserviert.
3. Zur Dimensionierung größerer Felder steht der DIM-Befehl, ebenfalls mit 2 Indizes, zur Verfügung.
4. Für mehrdimensionale DIM-Befehle und Feld-Variablen gelten dieselben Regeln wie für eindimensionale Felder.

Um Ihnen ein lehrreiches Beispiel für ein zweidimensionales Feld zu geben, in dem auch noch andere, für den C16 typische Anweisungen vorkommen, mache ich einen kurzen Ausflug in die Welt der Töne.

7. Hat man da Töne?

Im C16 sind zwei unabhängige Stimmen eingebaut. Mit beiden kann er singen wie ein Zeisig, mit der zweiten Stimme krächzt und zischt er zusätzlich wie eine heisere Katze. Mit diesen beiden Tongeneratoren lassen sich allerdings recht raffinierte Klangkombinationen erzielen.

Prinzipiell werden bei jedem tonerzeugenden Instrument drei Eigenschaften variiert:

- die Tonhöhe (T)
- die Lautstärke (L)
- die Tondauer (D)

Was für jede Flöte und jedes Saxophon gilt, gilt auch für die Stimmen des C16.

Mit dem Befehl *VOL*, einer Abkürzung des englischen Wortes »Volume« für Lautstärke, kann diese in 9 Stufen, von 0 bis 8, verändert werden.

Zur Festlegung von Tonhöhe und Tondauer gibt es einen zweiten Befehl *SOUND*, den ich eigentlich nicht zu übersetzen brauche, ist uns das Wort doch aus der heutigen Musik bekannt. Der SOUND-Befehl hat drei Variablen hinter sich stehen: mit der ersten werden die bereits genannten Tongeneratoren 1 oder 2 ausgewählt, die zweite Variable bestimmt die Tonhöhe mit Werten von 0 bis 1023, die dritte Variable gibt die Tondauer an und zwar von 1/50 Sekunde (0) bis 22 Minuten (65535). Ein schneller Direkt-Versuch zeigt es uns:

```
VOL 3:SOUND 1,500,50    (<RETURN>)
```

Aber lustiger und eindrucksvolller ist es, wenn sich bei Tönen etwas verändert. Dazu ein kleines Programm:

```
10 FOR L=0 TO 8
20 VOL L
30 SOUND 1,800+L*10,50
40 NEXT L
```

Sowohl die Lautstärke L in Zeile 20, als auch die Tonhöhe im 2. Ausdruck hinter dem SOUND-Befehl (mit L*10), werden in 9 Schritten verändert. Nur die Tondauer bleibt konstant auf 1 Sekunde (1/50 mal 50).

Der 2. Tongenerator wird mit SOUND 2,... ausgewählt. Er klingt, wie Sie sich leicht überzeugen können, genauso wie der erste. Aber Tongenerator 2 kann auch rauschen. Statt SOUND 2 braucht man nur SOUND 3 schreiben.

Daß es sich um nur zwei Tongeneratoren handelt, kann ich durch Einfügen einer weiteren Zeile, die den 2. Tongenerator einschalten soll, beweisen:

```
35 SOUND 2,700+L*10,50
```

Der Anfangston der 2. Stimme liegt durch den Wert 700 etwas tiefer. Nach RUN erklingen beide Stimmen gleichzeitig. Wenn Sie SOUND 2 durch SOUND 3 ersetzen, klingt eine Stimme und das Rauschen gleichzeitig. Wenn Sie aber in Zeile 30 SOUND 2 und in Zeile 35 SOUND 3 einsetzen, erklingt die Stimme und das Rauschen abwechselnd.

VOL- und SOUND-Befehl
- mit VOL, gefolgt von einer Zahl zwischen 0 und 8, wird die Lautstärke der Tongeneratoren eingestellt. Dabei gilt 0 für die kleinste, 8 für die größte Lautstärke.
- der VOL-Befehl muß vor dem ersten SOUND-Befehl gegeben werden, danach nur noch bei Veränderungen der Lautstärke.
- mit dem Befehl SOUND #,T,D wird ein Tongenerator eingeschaltet.
  # bestimmt den Tongenerator; mit 1 oder 2 erzeugt Tongenerator 1 oder 2 Töne, mit 3 erzeugt Tongenerator 2 nur Rauschen.
  T bestimmt die Tonhöhe; seine Werte liegen zwischen 0 und 1023. Die Zuordnung dieser Werte zu den Tonhöhen ist in Tabelle 2 dargestellt.
  D legt die Dauer des Tones fest und zwar mit Werten von 0 bis 65535. Diese Werte, mit 1/50 multipliziert, ergeben die Tondauer in Sekunden.
- die Töne können vor Ablauf der Tondauer D mit VOL 0 abgeschaltet werden.

Jetzt sind wir gerüstet, um eine Anwendung eines zweidimensionalen Feldes aufzubauen. Ich will Ihnen ein Programm zeigen, welches uns in recht einfacher Weise gestattet, eine Melodie zu komponieren und zu variieren.

8. Eine zweidimensionale Melodie

Als Melodie sollen uns dabei die ersten acht Takte der deutschen Nationalhymne dienen, die erstens allen bekannt sein dürfte, zweitens nur fünf verschiedene Tonhöhen hat, aber dafür drei Tondauern.

Ton	kleine Oktave	eingestrich. Oktave	zweigestrich. Oktave	dreigestrich. Oktave	viergestrich. Oktave	fünfgestrich. Oktave
A	7					
Ais/B	66					
H	118					
C	169	596	810	917	970	997
Cis/Des	220	620	822	923	973	999
D	262	643	834	929	977	1002
Dis/Es	300	664	844	934	979	1004
E	345	685	854	939	982	1006
F	383	704	864	944	984	1010
Fis/Ges	428	725	872	948	986	1011
G	453	739	881	953	988	1013
Gis/As	483	754	889	956	990	
A	516	770	897	960	992	
Ais/B	543	784	904	963	994	
H	571	798	911	967	996	

Tabelle 2. Zahlenwerte für Töne der chromatischen Tonleiter

Wir könnten nun natürlich einfach in neun Programmzeilen pro Ton einen SOUND-Befehl mit entsprechenden Variablen schreiben. Aber dann gibt uns das Programm nur diese eine Melodie, ohne die Möglichkeit, sie zu variieren.

Das erreichen wir, wenn wir die Töne und ihre Dauer mit dem INPUT-Befehl in ein zweidimensionales Feld plazieren, das dann wie folgt aussieht:

	M(x,0)	M(x,1)
1. Ton	(0,0)	(0,1)
2. Ton	(1,0)	(1,1)
3. Ton	(2,0)	(2,1)
8 .Ton	(7,0)	(7,1)
9 .Ton	(8,0)	(8,1)

Die Feldvariable M(x,0) soll für die Tonhöhe und M(x,1) für die Dauer eines Tones stehen.

Wir dimensionieren nun ein Feld für 9 mal 2 Eintragungen und zählen in einer Schleife die neun Töne der Melodie hoch. In jedem Durchgang der Schleife fragen wir per INPUT nach der Tonhöhe und der Dauer des jeweiligen Tones und nennen sie T und D:

```
10 DIM (8,1)
20 FOR I=0 TO 8
30 INPUT "TON,DAUER";T,D
40 M(I,0)=T
50 M(I,1)=D
60 NEXT I
```

Zwei Punkte sind dabei interessant:

Den ersten habe ich bereits genannt, er ist aber so wichtig, daß ich ihn wiederholen möchte: In den Feld-Variablen M(X,Y) bedeutet der erste Index die laufende Nummer des Tones, der zweite Index die Tonhöhe (0) oder die Dauer (1).

Der zweite Punkt taucht hier zum ersten Mal auf, nämlich die Eigenschaft des INPUT-Befehls, genau wie bei DIM oder READ, gleich mehrere Variable verarbeiten zu können. Im Programm müssen dann nach dem auffordernden Fragezeichen entsprechend viele Eingaben eingetippt werden.

Zusammenfassung INPUT-Befehl

1. Der INPUT-Befehl erlaubt, mehr als eine Variable festzulegen, denen dann über die Tastatur Daten zugewiesen werden. Die Variablen müssen durch Kommata voneinander getrennt sein:
INPUT "TEXT";A,B$,C,D
2. Die Variablen können gemischt, das heißt sowohl numerisch als auch Strings sein. Die Dateneingabe muß allerdings im Typ genau der Reihenfolge des INPUT-Befehls entsprechen.
3. Die Daten können sowohl einzeln, gefolgt von <RETURN>, als auch durch Kommata getrennt, alle in einer Zeile eingegeben werden. Im ersten Fall gibt der Computer vor jeder folgenden Eingabe zwei Fragezeichen aus.
4. Wenn mehr Daten eingegeben werden, als mit Variablen hinter dem INPUT-Befehl verlangt worden sind, erscheint die Meldung »?EXTRA IGNORED«, das heißt, der Computer ignoriert den Überschuß und macht normal weiter.

Das Programm wird fortgesetzt mit dem Spielen der Melodie:

```
70 FOR I=0 TO 8
80 VOL 3
90 SOUND 1,M(I,0),M(I,1)
100 NEXT I
```

Nach RUN erscheint die Aufforderung nach Tonhöhe und Dauer des ersten Tones. Wenn Sie Noten kennen, dann können Sie die Tonwerte der Tabelle 2 entnehmen - oder der Tabelle auf Seite 211 des Bedienungshandbuches von Commodore.

Ich empfehle Ihnen, mehrere Durchläufe zu machen, im ersten aber einen konstanten Wert für D, zum Beispiel 50, einzusetzen. Wenn Sie alle Werte eingegeben haben, erklingt die Melodie und Sie können hören, was Sie falsch gemacht haben.

Durch Eingabe von GOTO 70 erklingt die Melodie immer wieder.

Um Fehler zu korrigieren oder die richtige Dauer D einzugeben, fahren Sie mit dem Cursor auf den entsprechenden Wert und verändern die Zahlen, aber auf keinen Fall die <RETURN>-Taste drücken!

Wenn alles auf dem Bildschirm korrigiert ist, bringen Sie den Cursor auf das Wort RUN und drücken <RETURN>. Dadurch wird RUN wie vorher ausgeführt, und Sie müssen lediglich durch wiederholtes Drücken der <RETURN>-Taste die bereits korrigierten Werte »bestätigen«.

Wenn sie durchgelaufen sind, ertönt die korrigierte Melodie. Für die Noten-Unkundigen gebe ich die Zahlenwerte zum Eintippen an. Das soll Sie aber nicht hindern, ebenfalls wie oben beschrieben zu experimentieren.

Auf dem Bildschirm sollte stehen:

```
RUN
TON,DAUER? 810,60
TON,DAUER? 834,20
TON,DAUER? 854,40
TON,DAUER? 834,40
TON,DAUER? 864,40
TON,DAUER? 854,40
TON,DAUER? 834,20
TON,DAUER? 798,20
TON,DAUER? 810,40
```

Es gibt natürlich auch noch die konventionelle Methode, einen der SOUND-Werte zu ändern.

Wenn Ihnen zum Beispiel die Tonhöhe des dritten Tones zu tief ist, dann geben Sie einfach den neuen Wert direkt mit ein:

```
M(2,0)=860
```

Um die Dauer des letzten Tones zu verdoppeln, genügt die direkte Eingabe:

```
M(8,1)=80
```

Abschließend möchte ich Ihnen noch gern zeigen, wie man mit den beiden Tongeneratoren die obige Melodie zweistimmig erklingen läßt. Wir verwenden weiterhin das zweidimensionale Feld.

Der erste Index des Feldes bleibt gleich - er definiert weiterhin die Anzahl der Töne. Der zweite Index wird um 1 erhöht, denn zu der Tonhöhe und der Tondauer kommt jetzt noch die Nummer des Tongenerators hinzu. Wir nennen sie »Stimme« und geben der Variablen den Namen S. Natürlich wird das Feld wegen der zweiten Stimme doppelt so groß, was wir dem Programm mit dem DIM-Befehl der Zeile 10 mitteilen müssen:

```
10 DIM M(17,2)
20 FOR I=0 TO 17
30 INPUT "STIMME,TON,DAUER";S,T,D
35 M(I,0)=S
40 M(I,1)=T
50 M(I,2)=D
60 NEXT I
70 FOR I=0 TO 17
80 VOL 3
90 SOUND M(I,0),M(I,1),M(I,2)
100 NEXT I
```

In Zeile 30 wird wie vorher nach den Eingaben gefragt, nur sind es diesmal deren drei. Von den eingeschobenen Zeilen 35 bis 50 werden diese drei INPUTs den drei Feldvariablen zugewiesen und dann in Zeile 90 zum Erklingen gebracht. Die beiden Schleifen laufen jetzt entsprechend der Anzahl der Töne von 0 bis 17.

Um Ihnen die Sache zu erleichtern, gebe ich die ganze Folge der Töne an, die nach den Aufforderungen »STIMME,TON,DAUER?« wie folgt eingegeben werden müssen:

```
1,810,60        2,798,40
2,685,60        1,854,40
1,834,20        2,810,40
2,739,20        1,834,20
1,854,40        2,739,20
2,810,40        1,798,20
1,834,40        2,704,20
2,739,40        1,810,40
1,864,40        2,685,40
```

Diese Töne, oder genauer gesagt, das zweidimensionale Feld mit Variablen, welche unsere Töne repräsentieren, steht also im Speicher des C 16 gespeichert. Wenn wir den Computer ausschalten, wird der Speicher gelöscht und alles ist weg.

Es wäre deshalb schön, wenn wir das Feld mit den Tönen auch auf ein Band oder auf eine Diskette speichern könnten, um es bei späterer Gelegenheit wieder in den Computer holen zu können.

Aus dem Betriebshandbuch von Commodore (von Seite 34 bis 45) kennen Sie sicher die Speicher- und Ladebefehle SAVE, LOAD für Kassetten und DSAVE, DLOAD für Disketten.

Diese Befehle speichern und laden jedoch nur Programme, nicht aber Variablenwerte, die im Lauf eines Programms erst erzeugt werden wie zum Beispiel unsere Töne. Um also unsere Melodie mit diesen Befehlen auf Kassette oder Diskette speichern zu können, müßten wir die Zahlenwerte - pro Ton 1 SOUND-Befehl - im Programm festlegen. Es wäre dann allerdings kein Komponier-Programm mehr.

Zum Glück gibt es aber noch eine andere Methode, wie man Variablenwerte speichern kann. Dazu muß ich Ihnen aber erst ein bißchen mehr über den Speicher des C 16 erklären. Speichern Sie bitte dieses Fragment auf Band oder Diskette ab, wir werden es später vervollständigen.

Der nun folgende Abschnitt hat nur indirekt etwas mit Basic zu tun. Aber wie ganz am Anfang betont, komme ich ohne einige Beschreibungen der Eigenschaften oder besser gesagt Eigenheiten des C 16 nicht aus. Einige Kenntnisse des Innenlebens des Computers gehören eben auch zum Programmieren.

9. Das Gedächtnis des Computers

Alle Computer - Großrechenanlagen genauso wie kleine Heimcomputer - sind aus den folgenden Grundbausteinen aufgebaut:

- zentrale Recheneinheit, auch Mikroprozessor oder CPU (Central Prozessing Unit) genannt
- Speichereinheit
- Ein- und Ausgabe-Bausteine

Während die CPU rechnet, alle Vorgänge im Computer steuert, Befehle ausführt und somit das eigentliche Herz des

Computers darstellt, braucht man die Ein- und Ausgabe-Bausteine, um Daten in den Computer hinein- oder herauszuleiten. Alle Anschlüsse von Datasette, Diskettenlaufwerk, Drucker und Bildschirm werden von ihnen gesteuert. Der Speicher ist der Notizblock des Computers. In ihm steht alles, was er sich merken soll: Programmzeilen, Variablenwerte, Zeichenketten (Strings) und so weiter. Auf englisch heißt Speicher passenderweise Memory, was wir wiederum auf deutsch mit Gedächtnis übersetzen könnten.

Jeder Computer hat zwei Arten von Speichern:

Das *RAM* (**R**andom **A**ccess **M**emory) ist ein aus elektronischen Bauteilen aufgebauter Speicher, der für Programme frei verfügbar ist. Wir können in diesen Speicher Daten hineinschreiben und sie wieder herauslesen, ohne sie zu zerstören. Nur nach dem Ausschalten des Computers sind sie weg!

Das passiert auch dann, wenn nur ganz kurzzeitig der Strom ausfällt, etwa bei einem Gewitter oder wenn der Netzstecker einen Wackelkontakt hat.

Es ist deshalb ratsam, bei längeren Programmierarbeiten Zwischenergebnisse, das heißt den jeweiligen Inhalt des RAM auf Kassette oder Diskette abzuspeichern, denn nur dort sind sie dauerhaft sicher.

Das *ROM* (**R**ead **O**nly **M**emory) ist nicht frei verfügbar. Es besteht zwar auch aus elektronischen Bauteilen, aber sein Inhalt ist fest »eingebrannt« – man kann ihn nicht ändern. Er wird vom Computer selbst verwendet. Wenn Sie zum Beispiel Ihren C16 einschalten, dann läuft eine im ROM eingespeicherte Folge von Programmschritten ab, die schließlich mit der Meldung des Computers auf dem Bildschirm endet, mit der er sich bereit (READY) meldet. Im ROM stehen nicht nur die Programmschritte, die den Betrieb des Computers steuern, sondern auch das Programm, welches alle Basic-Befehle in einen Code »übersetzt«, den der zentrale Mikroprozessor (CPU) versteht.

10. Die Adressen der Speicherzellen

Der gesamte Speicher des C16 ist aus einzelnen Speicherzellen aufgebaut. Der C16 bietet Platz für 65536 Speicherzellen. Jede dieser Speicherzellen hat eine Nummer, von 0 bis 65535.

Diese Nummern nennen wir *Adressen*. Um eine Zahl oder ein Zeichen in eine bestimmte Speicherzelle hineinschreiben zu können, müssen wir ihre Adresse kennen. Natürlich gilt dasselbe für das Auslesen. Es liegt deshalb sehr nahe, uns ein »Adreßbuch« des Speichers zu beschaffen.

Im Betriebshandbuch ist eine derartige Liste im Anhang auf Seite 228 unter dem Titel »Speicherbelegung« angegeben. Diese Liste ist aber alles andere als klar und verständlich. In den Bildern 1 und 2 habe ich daher eine andere Darstellung gewählt, die Ihnen eine Übersicht über die verschiedenen Speicherbereiche und ihre Bedeutung geben soll. Aber ein Adreßbuch, oder wie es auf englisch heißt, eine »Memory Map«, ist das eigentlich auch nicht. Dazu müßte ich ja jede einzelne Adresse beschreiben.

Bild 1 ist also eine Kurzfassung eines Adreßbuches, in der wir aber bereits viel sehen können:

- Der Speicher beginnt ganz unten bei Adresse 0.
- Die ersten 4096 RAM-Speicherzellen werden vom Computer selbst benötigt.
- Ab Adresse 4096 beginnt der RAM-Speicher für Basic-Programme. Ab hier werden alle eingegebenen oder von Band und Diskette geladenen Programme gespeichert. Auch alle Variablen, Felder und Zeichenketten, die im Lauf eines Programms auftauchen, werden dort gespeichert.
- Bei einem C16 ohne Speichererweiterung endet dieser Speicherbereich bei Adresse 16383.

Theoretisch stehen dem Programmierer also 12287 Speicherplätze zur Verfügung. Warum sich der C16 nach dem Einschalten mit der Überschrift »12277 Bytes Free« meldet, und wo da zehn Speicherplätze abgezwackt werden, weiß ich leider auch nicht. Mein C16 meldet sich nämlich mit »60671 Bytes Free«, denn ich habe eine Speichererweiterung eingebaut, die den Programm-Speicher bis zur Adresse 64767 aufstockt. Bei dieser Erweiterung stimmt die Meldung des Computers mit der theoretisch verfügbaren Speicherzahl, nämlich mit der Differenz von 64767 minus 4096, überein. Wie dem auch sei:

- Beim C16 ohne Erweiterung, also in der Grundversion, ist der Bereich von 16384 bis 64767 leer; das heißt im Klartext, er enthält keine elektronischen Speicherschaltkreise.
- Ganz oben am Ende des Speichers hat der Computer wieder einige Plätze für seinen Eigenbedarf reserviert und zwar von 64768 bis 65535, das sind 768 Speicherplätze.

Alle diese Speicherzellen sind also RAM-Zellen, das heißt, man kann Daten hineinschreiben und herauslesen.

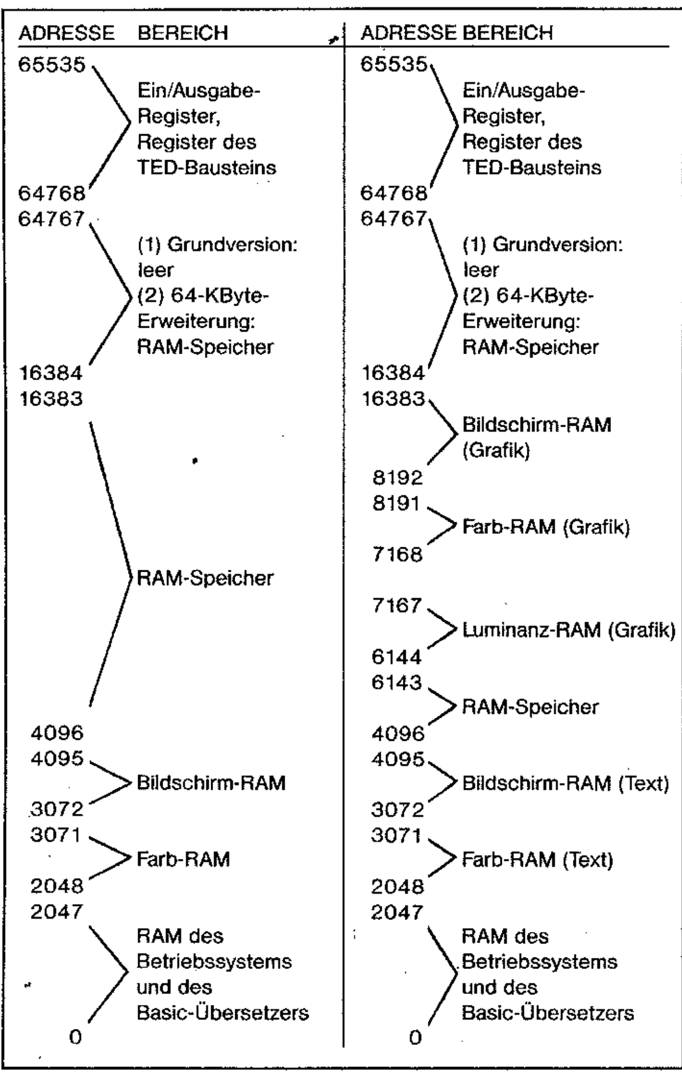


Bild 1. Speicher im Normal-Modus

Bild 2. Speicher im HiRes-Grafik-Modus

Wo, so werden Sie sicher fragen, ist der ROM-Speicher untergebracht? Des Rätsels Lösung habe ich Ihnen in Bild 3 aufgemalt: Von Adresse 32768 bis 64367 ist der Speicher zweistöckig; der ROM-Speicher sitzt dort auf denselben Adressen wie das RAM.

- von 32768 bis 53247 sind die fest vorgegebenen Programme zum Übersetzen von Basic in den Maschinen-Code enthalten
- anschließend bis 55295 sind alle Buchstaben und Zei-

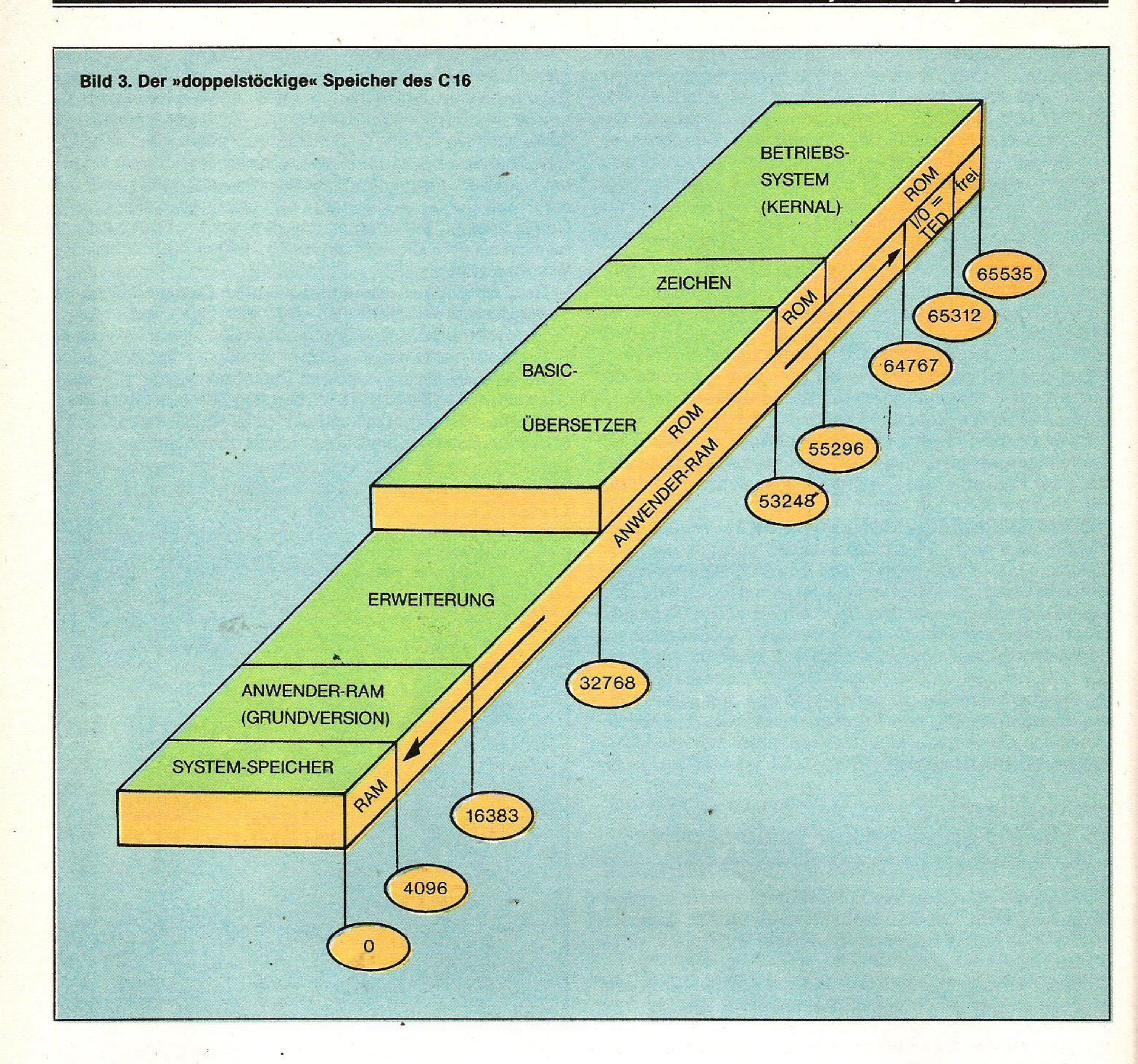


Bild 3. Der »doppelstöckige« Speicher des C 16

chen, die der C 16 kennt, gespeichert
- danach folgen die Programme des Betriebssystems.

Der Computer schaltet ganz einfach zwischen diesen Doppeladressen um, so wie es die Situation gerade erfordert.

Sie als Basic-Programmierer merken allerdings nichts davon, Sie arbeiten nur mit dem RAM-Bereich.

Später, wenn Sie einmal in das Gebiet der Maschinensprache vordringen, werden Sie den Umgang mit beiden Speicherbereichen lernen müssen.

Interessehalber will ich noch kurz auf Bild 2 eingehen:

Es zeigt die Speicheraufteilung des C 16 im sogenannten Grafik-Modus. Interessant ist dabei, daß in dieser Betriebsart dem Programmierer der Speicherbereich von 6144 bis 16383 weggenommen wird, so daß in der Grundversion des C 16 nur der Bereich von 4096 bis 6143 als Programmspeicher bleibt - und das ist herzlich wenig!

Vorläufig bleiben wir beim Normal-Modus, also bei den Verhältnissen, die in Bild 1 gezeigt sind.

Uns stellt sich als nächstes die Frage, wie wir Daten in einzelne Speicherzellen des Programmspeichers hineinschreiben und herauslesen können.

11. Im Speicher stöbern

In Basic gibt es zwei Befehle, mit deren Hilfe der Speicher abgefragt werden kann.

Während der Lese-Befehl völlig unschädlich ist - er »kopiert« praktisch den Inhalt der Speicherzelle und läßt das Original unverändert - kann der Schreib-Befehl gefährlich sein. Ich will Sie gleich zu Beginn davor warnen, ihn unbedacht einzusetzen - er verändert den Inhalt einer Speicherzelle, und über die Folgen dieser Veränderung für ein Programm oder den Betrieb des Computers muß man sich im klaren sein.

Der Schreib-Befehl heißt *POKE*, was das englische Wort für »hineinstochern« ist. Nach dem Befehlswort muß die Adresse der Speicherzelle stehen, danach folgt, durch ein Komma getrennt, die Zahl, die hineingeschrieben werden soll.

Der Befehl

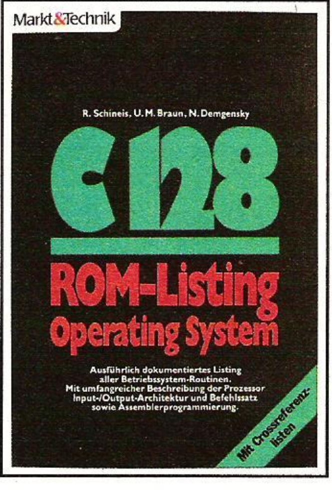
```
POKE 14000,55
```

schreibt also die Zahl 55 in die Speicherzelle 14000.

Bücher zu AMIGA/C128

M. Breuer
Das AMIGA-Handbuch
März 1986, 461 Seiten

Der Commodore AMIGA stellt einen neuen Schritt in der Entwicklung der Personal Computer dar. Er setzt die neuesten Entwicklungen der Chip-Technologie ein, um dem Endanwender eine extrem leistungsfähige Maschine zu einem vergleichsweise günstigen Preis auf den Schreibtisch stellen zu können. Der AMIGA besitzt enorme Farbgrafik-Fähigkeiten, die auch für die Benutzerführung konsequent eingesetzt werden.
Das Buch liefert übersichtlich gegliedertes Grundwissen über die neue Commodore-Maschine. Aus dem Inhalt: Vorhang auf: Der AMIGA! · Auf der Werkbank des AMIGA · Grundlage der Bedienung des AMIGA · Grafik mit Graficraft und Delux Paint. AMIGA für Fortgeschrittene: Das CLI · Automatisierung des AMIGA · Die Spezialchips des AMIGA · Grundlagen von Sound und Grafik.
• Mit vielen Abbildungen und Übersichtstafeln für den täglichen Einsatz.
Best.-Nr. MT 90228
ISBN 3-89090-228-6
DM 49,–/sFr. 45,10/öS 382,20



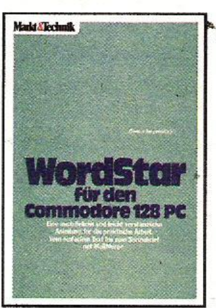
R. Schineis, M. Braun, N. Demgensky
C128-ROM-Listing: Operating System
März 1986, 450 Seiten

Dieses Buch ist für alle Programmierer und Anwender gedacht, die mehr über ihren Commodore 128 PC wissen wollen. Ein umfangreiches, vollständig kommentiertes Assemblerlisting mit Cross-Referenzliste (Verweistabelle) umfaßt das komplette Betriebssystem mit dem 40/80-Zeichen-Editor, des eingebauten Maschinensprache-Monitors sowie allen Kernal-Routinen.
Best.-Nr. MT 90221
ISBN 3-89090-221-9
DM 49,–/sFr. 45,10/öS 382,20

R. Schineis, M. Braun
C128-ROM-Listing: BASIC-7.0-Betriebssystem
3. Quartal 1986, ca. 300 Seiten

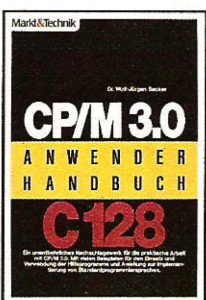
Eine umfassende Beschreibung des BASIC-Interpreters. Mit vollständig kommentiertem Assemblerlisting und Cross-Referenzliste.
Best.-Nr. MT 90220
ISBN 3-89090-220-0
DM 49,–/sFr. 45,10/öS 382,20

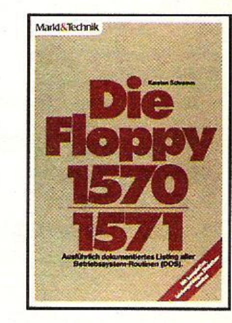












M. Kohlen
Grafik auf dem AMIGA
3. Quartal 1986, ca. 250 S.

Dieses Buch setzt sich mit den außerordentlichen Grafikfähigkeiten des AMIGA auseinander. Es enthält zum einen eine ausführliche Beschreibung der Grafikhard- und -software des AMIGA und ihrer Funktionsweise. Zum anderen will es aber auch in die Grundzüge der Grafikprogrammierung überhaupt einführen. In zwei Einleitungskapiteln wird, diese Informationen in einer für den unvorbereiteten Leser verständlichen Form vermittelt. In den folgenden Kapiteln werden diese Kenntnisse dann in praktischen Beispielen umgesetzt. Außerdem bietet das Buch einen Überblick über die Soft- und Hardwareerweiterungen für den AMIGA.
Best.-Nr. MT 90236
ISBN 3-89090-236-7
DM 49,–/sFr. 45,10/öS 382,20

G. Jürgensmeier
WordStar 3.0 mit MailMerge für den Commodore 128 PC
1985, 435 Seiten
Best.-Nr. MT 780
ISBN 3-89090-181-6
DM 49,–/sFr. 45,10/öS 382,20

Dr. P. Albrecht
dBASE II für den Commodore 128 PC
1985, 280 Seiten
Best.-Nr. MT 838
ISBN 3-89090-189-1
DM 49,–/sFr. 45,10/öS 382,20

Dr. P. Albrecht
Multiplan für den Commodore 128 PC
1985, 226 Seiten
Best.-Nr. MT 836
ISBN 3-89090-187-5
DM 49,–/sFr. 45,10/öS 382,20

G. Möllmann
C128-Programmieren in Maschinensprache
3. Quartal 1986, ca. 250 Seiten

Ein Buch, das alle Informationen bietet, um erfolgreich auf dem C128 zu programmieren. Dazu gehört auch der Umgang mit den ROM-Routinen aus Basic und Betriebssystem.
Best.-Nr. MT 90213
ISBN 3-89090-213-8
DM 52,–/sFr. 47,80/öS 405,60

P. Rosenbeck
Das Commodore 128-Handbuch
1985, 383 Seiten

Dieses Buch sagt Ihnen alles, was Sie über Ihren C128 wissen müssen: die Hardware, die drei Betriebssystem-Modi und was die CP/M-Fähigkeit für Ihren Computer bedeutet.
Best.-Nr. MT 90195
ISBN 3-89090-195-6
DM 52,–/sFr. 47,80/öS 405,60

H. Ponnath
Grafik-Programmierung C128
März 1986, 196 Seiten, inkl. Disk

Die Programmierung von Grafik gehört zu den interessantesten Aufgaben, die man mit dem Commodore 128 PC lösen kann. Dieses Buch hilft Ihnen dabei! Das Themenfeld ist weit gespannt und behandelt unter anderem: hochauflösende- und Mehrfarben-Grafik im C 128-Modus.
Best.-Nr. MT 90202
ISBN 3-89090-202-2
DM 52,–/sFr. 47,80/öS 405,60

J. Hückstädt
BASIC 7.0 auf dem Commodore 128
1985, 239 Seiten

An praxisnahen Beispielen zeigt dieses Buch, wie man die für den 128er typischen Merkmale und Eigenschaften (Sprites, Shapes, hochauflösende Grafik) optimal nutzt.
Best.-Nr. MT 90149
ISBN 3-89090-149-2
DM 52,–/sFr. 47,80/öS 405,60

J. Hückstädt
CP/M-3.0-Anwender-Handbuch C128
Mai 1986, 250 Seiten

Wenn Sie Ihren Commodore 128 PC schon ganz gut im Griff haben und jetzt so richtig einsteigen wollen in die Möglichkeiten, die das leistungsstarke Betriebssystem CP/M-3.0 bietet, sollten Sie mal in dieses Buch schauen: Es sagt Ihnen alles über den Aufbau einer Datenverarbeitungsanlage, Mikrocomputer, Programmiersprachen und Betriebssysteme im allgemeinen und über das Betriebssystem CP/M speziell auf dem C128.
Best.-Nr. MT 90196
ISBN 3-89090-196-4
DM 52,–/sFr. 47,80/öS 405,60

K. Schramm
Die Floppy 1570/1571
Mai 1986, 470 Seiten

In der Floppy 1571 wurde ein völlig neues Floppy-Konzept verwirklicht: Diese Floppystation ist in der Lage, mehrere verschiedene Diskettenformate zu verarbeiten.
Dieses Buch soll es sowohl dem Einsteiger als auch dem fortgeschrittenen Programmierer ermöglichen, die vielfältigen Möglichkeiten dieses neuen Gerätes voll auszuschöpfen.
Best.-Nr. MT 90185
ISBN 3-89090-185-9
DM 52,–/sFr. 47,80/öS 405,60



Fragen Sie Ihren Buchhändler nach unserem kostenlosen Gesamtverzeichnis mit über 200 aktuellen Computerbüchern und Softwareprogrammen. Oder fordern Sie es direkt beim Verlag an!

Markt&Technik-Fachbücher erhalten Sie bei Ihrem Buchhändler

Bestellungen im Ausland bitte an den Buchhandel oder an untenstehende Adressen.
Schweiz: Markt&Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, ☎ 042/41 56 56
Österreich: Ueberreuter Media Handels- und Verlagsges. mbH, Alser Straße 24, 1091 Wien, ☎ 0222/48 15 38-0

Irrtümer und Änderungen vorbehalten.



Unternehmensbereich Buchverlag
Hans-Pinsel-Straße 2, 8013 Haar bei München

POKE-Befehl
- er wird in der folgenden Weise geschrieben:
POKE Adresse,Wert
Adresse und Wert müssen durch ein Komma voneinander getrennt sein
- der POKE-Befehl speichert den »Wert« in der mit »Adresse« bezeichneten Speicherzelle. Er überschreibt dabei einen dort gespeicherten früheren Wert.
- gültige »Werte« liegen im Bereich von minimal 0 bis maximal 255. Wird dieser Bereich überschritten, erscheint die Fehlermeldung »ILLEGAL QUANTITY ERROR«.
- der erlaubte Bereich für »Adresse« reicht von 0 bis 65535. Ein Überschreiten erzeugt wiederum die Fehlermeldung »ILLEGAL QUANTITY ERROR«.

Der Lese-Befehl heißt *PEEK*, das englische Wort für »hineinschauen«. Hinter diesem Befehlswort steht die Adresse – und zwar in Klammern! – deren Inhalt gelesen werden soll. Die Befehlsfolge:

```
A=PEEK(14000):PRINT A
```

oder noch kürzer

```
PRINT PEEK(14000)
```

liest den Inhalt der Speicherzelle 14000 und druckt ihn auf den Bildschirm.

In der ersten Zeile wird der Inhalt der Variablen A zugeordnet und steht unter diesem Namen für spätere Verwendung im Speicher. Da der Inhalt aber trotz PEEKen in der Zelle 14000 stehenbleibt und von dort immer wieder herausgeholt werden kann, bietet sich die Kurzform der zweiten Zeile an, die direkt das PEEK-Ergebnis ausdruckt.

PEEK-Befehl
- er wird in der folgenden Weise geschrieben:
PEEK (Adresse)
- die »Adresse« muß immer in Klammern stehen.
- der Befehl liest den Inhalt der durch die »Adresse« angegebenen Speicherzelle.
- der erlaubte Bereich von »Adresse« reicht von 0 bis 65635. Wird er überschritten, meldet der Computer dies mit »ILLEGAL QUANTITY ERROR«.

Wir haben gerade vorhin mit dem POKE-Befehl die Zahl 55 in die Speicherzelle 14000 hineingeschrieben und sie danach mit PEEK wieder ausgelesen.

Nun, das beweist natürlich noch gar nichts. Schauen wir also mal im Speicher rund um die Zelle 14000 nach, was da so drin steht. Mit den Programmzeilen:

```
10 FOR I=13900 TO 14200
20 PRINT I,PEEK(I)
30 NEXT I
```

schauen wir uns den Speicherbereich von 13900 bis 14200 an. Durch Zeile 20 wird der jeweilige Wert von I und daneben die Zahl, die in der Adresse I gespeichert ist, mit nur einem PRINT-Befehl ausgedruckt. Das Komma zwischen den beiden bewirkt den Abstand von einer Viertel-Zeilenlänge.

Nach RUN sehen wir, daß in allen Speicherzellen entweder eine 0 oder 255 steht, mit Ausnahme der Zelle 14000, da steht unsere 55. Übrigens hoffe ich, daß Sie wissen, daß mit der Commodore-Taste links unten der Ablauf des Programms gebremst werden kann – zum besseren Überblick, wenn die Speicherzelle 14000 vorbeisaust. In diesem Teil des Speichers steht also praktisch gar nichts. Das ist auch kein Wunder, denn wir tummeln uns ja im oberen Teil des RAM-Speichers, der uns für Basic-Programme zur Verfügung steht. Unser Mini-Programm von drei Zeilen reicht da natürlich bei weitem nicht hin. Wenn Sie Zeile 20 so abändern, daß der Ausdruck ab dem Speicheranfang 4096 beginnt:

```
20 FOR I=4096 TO 5000
```

dann sehen wir in der Tat ein Durcheinander von Zahlen, die bis hin zur Adresse 4150 reichen. Das ist – in einem besonderen Code geschrieben – unser Programm. Danach kommen wieder die Leerserien mit 0 und 255.

Wenn Sie mehr über Code und Schreibweise, mit denen die Programmzeilen im RAM abgespeichert werden, erfahren wollen, dann lesen Sie bitte den Aufsatz »Den C 16 und VC 20 durchschaut« von Christoph Sauer im 64'er-Sonderheft 3/1986, das ich bereits in der Einleitung als wertvolle Fundgrube bezeichnet habe. In diesem Aufsatz finden Sie ab Seite 32 im Text und in Bild 2, Tabelle 2 und Bild 5b entsprechende Erklärungen.

Schauen wir spaßeshalber noch in den obersten Teil des Speichers, wo laut Bild 1 die mysteriösen Register für Ein- und Ausgabe und für den TED-Baustein liegen. Ich wähle Speicherzelle 65287 als Versuchskaninchen.

```
PRINT PEEK (65287)
```

Das ergibt eine 8.

Jetzt machen wir ein Experiment – das, wie gesagt, unerwünschte Folgen haben kann. Wir ändern mutwillig den Inhalt dieser Speicherzelle mit:

```
POKE 65287,0
```

Und siehe da, nach Drücken der <RETURN>-Taste verschiebt sich der linke Rand des Bildschirms und schneidet alle linken Zeichen ab. Sie sind zwar noch da, aber nicht sichtbar.

Der Originalzustand läßt sich mit dem oben ermittelten »Normalwert« wieder herstellen, indem Sie – am Anfang blind – eintippen:

```
POKE 65287,8
```

Dieser Versuch ist gutgegangen. Aber wenn Sie statt der 0 oder der 8 die Zahl 33 in die Zelle POKEn:

```
POKE 65287,33
```

dann geht es schief. Der Bildschirm wird dunkel und durch keine Steuertasten läßt er sich wieder beleben – der Computer ist »abgestürzt«!

Ein Mittel bleibt uns doch, nämlich die Reset-Taste, die sich winzig klein neben dem Netzschalter auf der rechten Seite des C 16 befindet.

Zusammenfassung POKE-Befehl
1. Der POKE-Befehl ist nützlich und gefährlich zugleich.
2. Wird eine bestimmte Zahl in eine Speicherzelle gePOKEt, mit der der Computer seine eigenen Abläufe steuert (Bereiche 0 bis 4095, 64768 bis 65535), kann dadurch der Ablauf beeinflußt werden. Es kann aber auch zum »Absturz« des Computers führen.
3. Mit »Absturz« wird beim Computer der Zustand bezeichnet, in dem kein Programm mehr läuft und der Computer auf keine normalen Steuertasten mehr reagiert.
4. Ein abgestürzter Computer kann nur durch Aus- und Wiedereinschalten oder durch die Reset-Taste wieder in Gang gesetzt werden.
Nach dem Aus- und Einschalten befindet sich der Computer im Anfangszustand, das heißt, ein Programm, das vorher im Arbeitsspeicher war, ist verloren.
Mit der Reset-Taste geht der Computer ebenfalls in den Anfangszustand zurück, aber der Speicherinhalt bleibt erhalten. Mit einem Trick, der im Abschnitt »Erste Hilfe – Basic-Programme retten nach NEW oder Reset« beschrieben ist, kann es wieder hergestellt werden.
5. Es ist empfehlenswert, vor einem POKE-Befehl ein im Arbeitsspeicher befindliches Programm für alle Fälle zuerst auf Band oder Diskette zu speichern!

12. Erste Hilfe – Basic-Programme retten nach NEW oder Reset

Diese Überschrift stammt nicht von mir, sondern aus dem schon zitierten 64'er-Sonderheft 3/1986, wo auf Seite 32 beschrieben wird, wie der folgende Trick funktioniert.

Da er Kenntisse des Betriebssystems des Computers und eines Befehls (SYS), den wir nicht behandeln werden, vorausetzt, gebe ich den Trick nur als Kochrezept an: Voraussetzung: Sie haben ein Programm im Computer, zum Beispiel:

```
10 PRINT A
20 PRINT B
30 PRINT C
```

Jetzt geben Sie - natürlich aus Versehen - NEW ein oder - mit Absicht - den vorher ausprobierten Absturz-POKE-Befehl POKE 65287,33.

Im zweiten Fall drücken Sie die Reset-Taste.

In beiden Fällen können Sie versuchen, die drei Programmzeilen zu LISTen - das Programm ist weg!

Geben Sie jetzt das Kochrezept direkt ein:

```
POKE4097,1: DELETE 1
```

Nach <RETURN> ist das Programm wieder da - mit dem LIST-Befehl leicht nachprüfbar.

13. Wir POKEn noch ein Weilchen

Es gibt keine Computer-Zeitschrift, die nicht unter der Rubrik Tips und Tricks alle möglichen POKE-Adressen angibt, mit denen sich verblüffende Effekte erzielen lassen. Für den C 16 sind diese Tips noch recht selten. Ich gebe Ihnen daher ein paar Hinweise.

```
POKE 194,1:PRINT "ABCDE"
```

druckt alle Zeichen dieser Programmzeile revers (invertiert).

```
POKE 1344,64
```

schaltet die Wiederholfunktion aller Tasten aus

```
POKE 1344,0
```

nur die Leer-, INST/DEL- und alle Cursor-Tasten wiederholen, solange sie gedrückt werden

```
POKE 1344,128
```

alle Tasten haben Wiederholfunktion (Normalzustand)

```
POKE 2026,255
```

Zeichen werden nicht überschrieben, sondern eingefügt

```
POKE 2026,0
```

Zeichen werden überschrieben (Normalzustand)

```
POKE 65301,20
```

schaltet die Farbe des Bildschirm-Hintergrundes auf violett

```
POKE 65305,10
```

schaltet die Farbe des Bildschirm-Rahmens auf grün.

Die beiden letzten POKE-Adressen 65301 und 65305 wollen wir uns näher anschauen.

Der Zahlenwert in der Speicherzelle 65301 bestimmt also die Hintergrundfarbe, während in Speicherzelle 65305 die Rahmenfarbe festgelegt ist. Da nach einem POKE-Befehl Zahlen von 0 bis 255 zugelassen sind, ist es sicher ganz interessant, welche Zahl welche Farbe hervorruft. Ein kleines Programm gibt uns darüber Auskunft:

```
10 FOR I=0 TO 255
20 POKE 65301,I
30 PRINT I
40 GET A$:IF A$="" THEN 40
50 NEXT I
```

Zwischen den Zeilen 10 und 50 wird in einer Schleife die Variable I von 0 bis 255 hochgezählt.

Der jeweilige Wert von I wird in Zeile 20 in die Speicherzelle 65301 gePOKEt und ändert dadurch die Hintergrundfarbe.

Um die Zugehörigkeit der Farben zu den Zahlen zu sehen, wird in Zeile 30 der jeweilige Wert von I ausgedruckt.

Zeile 40 dient dazu, die Schleife schrittweise weiterzuschalten. Der GET-Befehl in dieser Zeile springt solange auf seine eigene Zeilennummer zurück, bis irgendeine beliebige Taste gedrückt wird. Erst dann kommt der NEXT-Befehl in Zeile 50 zum Zug.

Diese Anwendung des GET-Befehls habe ich im Anfangskurs in Sonderheft 5/1986 auf Seite 56 vorgestellt.

Mit diesem kleinen Programm werden pro Tastendruck alle Farben »durchgeleiert«, wobei die jeweils als unterste ausgedruckte Zahl dem Farbwert entspricht.

Das Basic des C 16 weist einen speziellen Befehl auf, der die Zeile 40 wesentlich vereinfacht.

Er heißt *GETKEY* und erfüllt denselben Zweck wie die ganze Befehlsfolge der Zeile 40. Diese lautet nun:

```
40 GETKEY A$
```

So einfach kann Programmieren sein.

GETKEY-Befehl

- der Befehl wird so geschrieben: GETKEY Variable.
- GETKEY weist das Zeichen einer beliebig gedrückten Taste der Variablen zu. Dabei muß der Typ der gedrückten Taste mit dem Variablentyp hinter GETKEY übereinstimmen.
- die Programmzeile: 10 GETKEY A$
  ist identisch mit: 10 GET A$:IF A$="" THEN 10
- im Unterschied zu GET wartet GETKEY solange, bis eine Taste gedrückt worden ist.

Was wir vorher mit der Speicherzelle 65301 für die Hintergrundfarbe gemacht haben, können wir ebenso mit der Zelle 65305 für die Umrandung machen.

Diese beiden Adressen gehören zu den »Registern« des TED-Bausteins, der für Töne und Grafik zuständig ist.

Zusammenfassung Register

1. Ein Register ist eine Speicherzelle im Mikroprozessor (CPU) oder in einem anderen elektronischen Baustein. Im C 16 besteht ein Register aus 8 Bit, das entspricht 1Byte.
2. In einem Register werden Daten gespeichert, die den Ablauf von arithmetischen, logischen oder von Steueroperationen festlegen.
3. Der Inhalt von Registern kann mit PEEK ausgelesen und was mit POKE verändert werden.

Jetzt kennen wir also die beiden Farbregister und wissen, wie wir die Farben des Bildschirms unseren Wünschen anpassen können.

Bevor Sie jetzt anfangen, sich mit dem letzten kleinen Programm eine Liste aller Farbcodes anzufertigen, will ich Ihnen schnell sagen, daß auch in diesem Fall das Basic des C 16 einen Befehl zur Verfügung stellt, der ein langwieriges POKEn in die beiden Farbregister unnötig macht.

Dieser Befehl heißt *COLOR*.

Ihm werden drei Parameter beigegeben, für den Farbbereich, für die Farbe und für die Helligkeit, die auch »Luminanz« genannt wird. Nähere Einzelheiten stehen in der folgenden Befehlserklärung.

Für uns kommen zur Zeit nur die Farbbereiche 0, 1 und 4 in Frage.

Farbbereich 2 und 3 bieten die Möglichkeit, im sogenannten Grafik-Modus allen Zeichen und Buchstaben zwei Farben mehr, also 3 verschiedene Farben, zu geben. Ich gehe in diesem Kurs auf diesen Modus nicht ein. Wenn Sie aber etwas über »Multicolor« wissen wollen, dann lesen Sie bitte den Aufsatz »Grafik und Sound mit dem C16« von C. Spitzner im schon oft zitierten 64'er-Sonderheft 3/1986 ab Seite 21.

14. Der C16 hat 121 Farben

Mit dem COLOR-Befehl können wir also 16 x 8 verschiedene Farben einstellen. Aber Sie werden sehen, in Wirklichkeit sind es nur 15 x 8+1, nämlich 121, weil es von der Farbe schwarz nur eine einzige Helligkeitsstufe gibt.

COLOR-Befehl

- er wird mit drei Angaben versehen:
  COLOR Farbbereich,Farbe,Helligkeit
- mit Zahlen von 0 bis 4 können 5 Farbbereiche ausgewählt werden und zwar:
  0 Bildschirm-Hintergrund
  1 Buchstaben und Zeichen (Vordergrund)
  2 Mehrfarben-Modus 1
  3 Mehrfarben-Modus 2

4 Bildschirm-Umrandung
- die Farbe des ausgewählten Farbbereiches wird durch 16 Zahlenwerte festgelegt. Es gilt:

1 schwarz	9 orange
2 weiß	10 braun
3 rot	11 gelbgrün
4 lila	12 rosa
5 purpur	13 blaugrün
6 grün	14 hellblau
7 blau	15 dunkelblau
8 gelb	16 hellgrün

- mit Zahlen von 0 bis 7 können für jede der oben genannten Farben (außer für schwarz) 8 Helligkeitsstufen (Luminanz) eingestellt werden. 0 ist die dunkelste, 7 ist die hellste Stufe.

Der Wert für die Helligkeit kann auch weggelassen werden. Dann wird er automatisch auf den Normalwert 6 gesetzt.

Wie üblich wollen wir uns ein kleines Übungsprogramm entwickeln, mit dem wir diesmal alle Farbkombinationen in allen drei Farbbereichen ausprobieren können.

Im Mittelpunkt steht natürlich der COLOR-Befehl:

```
70 COLOR B,F,H
```

Den Bereich B wollen wir mit der 0-, 1- und 4-Taste auf 0 (=Hintergrund), 1 (=Zeichen) und 4 (=Umrandung) umschalten.

Die Farbe F und die Helligkeit H wollen wir durch mehrfaches Drücken der entsprechenden »Bereichs«-Taste weiterschalten. Die Abfrage einer Taste geht mit GETKEY wie gehabt. Um zu prüfen, ob eine der drei genannten Tasten gedrückt worden ist, gibt es mehrere Möglichkeiten. In Zeile 40 steht das Zeichen der Taste als String zwischen Anführungszeichen.

```
30 GETKEY A$
40 IF A$="0" THEN B=0
```

Wir können aber auch die ASCII-Codewerte der Tasten abfragen, die wir der ASCII-Tabelle des Bedienungshandbuches entnehmen. Wir wollen das mit der 1-Taste für den Bereich 1 machen:

```
50 IF A$=CHR$(49) THEN B
50 IF A$=1:PRINT"A";
```

Wenn B auf 1 gesetzt wird, ist der Farbbereich auf »Zeichen« (Vordergrund) geschaltet. Deswegen drucken wir in diesem Modus den Buchstaben A, mit Semikolon nebeneinander gesetzt, aus.

Die dritte Abfragetechnik verwendet ebenfalls den ASCII-Code, allerdings in Verbindung mit dem schon bekannten ASC-Befehl (Seite 60 im Anfangskurs).

```
60 IF ASC(A$)=52 THEN B=4
```

Um F und H laufend zu verändern, zählen wir beide in zwei geschachtelten Schleifen hoch

```
10 FOR F=1 TO 16
20 FOR H=0 TO 7
80 NEXT H,F
90 GOTO 10
```

Der Rücksprung von Zeile 90 sorgt für Wiederholbarkeit des ganzen Vorganges.

Listing 2 bringt alle Zeilen in die richtige Reihenfolge:

```
10 FOR F=1 TO 16
20 FOR H=0 TO 7
30 GETKEY A$
40 IF A$="0" THEN B=0
50 IF A$=CHR$(49) THEN B=1:PRINT"A";
60 IF ASC(A$)=52 THEN B=4
70 COLOR B,F,H
80 NEXT H,F
90 GOTO 10
```

Sooft Sie die <0> drücken, schalten Sie die Hintergrundfarbe durch alle ihre Helligkeitsstufen weiter. Die <1>-Taste macht dasselbe mit dem Buchstaben A, der allerdings immer wieder neu auf den Bildschirm geschrieben wird. Die <4>-Taste schließlich erlaubt Ihnen, die Farbe der Umrahmung weiterzuschalten.

Mit dem COLOR-Befehl ersetzen wir also auf einfache Weise das POKEn der Farbregister. Was tun wir aber, wenn wir die Farbe und die Helligkeit der gerade eingestellten Farbbereiche ermitteln wollen? Nun, wir könnten mit dem PEEK-Befehl in den Registern nachschauen. Das geht in der Tat, aber auch hierfür bietet uns das Basic des C 16 schnellere Alternativen.

Mit dem Befehl *RCLR*, abgekürzt aus »Return Color«, was soviel heißt wie »Farbe zurückholen«, können wir die Zahl der gerade eingestellten Farbe erfragen.

Entsprechend holt uns der Befehl *RLUM* den Helligkeitswert (Luminanz).

Zusammenfassung RCLR- und RLUM-Befehl
- RCLR (B) gibt den dem Farbbereich B (0 bis 4) zugeordneten Farbwert an
- RLUM (B) gibt die dem Farbbereich B zugeordnete Helligkeit an
- bei beiden Befehlen muß das Ergebnis entweder einer Variablen zugeordnet werden, oder es wird in Verbindung mit einem anderen Befehl (zum Beispiel PRINT) verwendet.

Wenn Ihnen beim Verwenden des Listing 2 eine Farbkombination besonders gut gefällt, können Sie die Werte im Direkt-Modus herausfinden:

```
PRINT "HINTERGRUND" RCLR(0); RLUM(0)
PRINT "ZEICHEN" RCLR(1);RLUM(1)
PRINT "UMRANDUNG"RCLR(4);RLUM(4)
```

Ein bißchen mit den Farben experimentieren lohnt schon, denn nicht alle Farben passen zueinander und nicht alle Buchstabenfarben sind auf allen Hintergrundfarben gut zu lesen. Als Regel gilt, daß Hintergrund- und Zeichenfarbe derselben Helligkeit nicht lesbar sind.

15. Der Bildschirmspeicher

Ich möchte Sie gern noch ein bißchen länger mit dem Speicher beschäftigen. Schauen Sie sich bitte nochmal Bild 1 an. Da sehen wir ab Adresse 3072 bis 4095 das »Bildschirm-RAM«.

Um zu demonstrieren, was das ist, zeige ich Ihnen ein kleines Experiment:
- Setzen Sie den Computer mit einem Reset zurück
- Löschen Sie den Bildschirm mit der <CLEAR>-Taste
- Fahren Sie mit dem Cursor in die untere Hälfte des Bildschirms
- Geben Sie direkt ein:

```
POKE 3072,1
```

Ganz links oben auf dem Bildschirm steht plötzlich ein A.

Wenn Sie aus der 1 eine 2 machen und den Befehl wiederholen, verwandelt sich das A in ein B.

Und noch ein Versuch: ändern Sie 3072,2 in 3073,3 um und geben es ein. Jetzt steht neben dem B ein C.

Die Zahl 1 erzeugt also ein A, 2 ein B und 3 ein C. Dann müßte eigentlich die 26 ein Z hervorrufen – was sie auch tut.

Wir haben also einen neuen Code für die Zeichen auf dem Bildschirm gefunden.

Ein ähnlicher Zusammenhang deutet sich bei den Adressen an: 3072 setzt den Buchstaben an den ersten Platz des Bildschirms, 3073 an den zweiten. Der Bildschirm hat 40 Stellen pro Zeile und das für 25 Zeilen. Das macht insgesamt 1000 Plätze auf dem Bildschirm.

Daher müßte die Adresse 4071 einen Buchstaben ganz rechts unten plazieren.

```
POKE 4071,25
```

setzt ein Z genau dorthin, wie vorhergesagt.

Ich will Sie nicht länger plagen und das alles zusammenfassen:



Bildschirmspeicher

1. Der Elektronenstrahl, der mit großer Geschwindigkeit über den Bildschirm des Fernsehers oder des Monitors flitzt und dort Bilder oder Text hinmalt, hat kein Gedächtnis. Deswegen muß der Computer alle Angaben, die der Elektronenstrahl braucht, in einem gesonderten Speicher festhalten, der deshalb »Bildschirmspeicher« heißt.
2. Im Bildschirmspeicher sind alle Zeichen gespeichert, die zum jeweiligen Zeitpunkt auf dem Bildschirm erscheinen.
3. Für den Bildschirmspeicher sind im RAM die Speicherzellen 3072 bis 4095 reseviert.

Da auf dem Bildschirm des C16 genau 1000 Plätze vorhanden sind (40 Stellen mal 25 Zeilen), reicht der Bildschirmspeicher nur bis Adresse 4071. Die restlichen 24 Byte sind frei.

4. Alle Zeichen und Buchstaben stehen im Bildschirmspeicher mit einem speziellen Code, der im Bedienungshandbuch auf den Seiten 213 und 214 aufgelistet und auf Seite 212 erklärt ist. Der Bildschirmcode hat nichts mit dem ASCII-Code zu tun.

Durch direktes POKEn von Bildschirm-Code-Werten in den Bildschirmspeicher lassen sich Effekte erzielen, die mit dem PRINT-Befehl nur sehr umständlich möglich wären. Ich will Ihnen das an einem Beispiel zeigen.

16. Bildschirmeffekte

Ziel des Experiments soll sein, eine bewegte farbige Umrandung des Bildschirms zu erzeugen, die als Programmteil in anderen Programmen verwendbar ist. Ich verwende dazu ein Beispiel aus dem Buch »VC 20-Spielebuch« von A. Dripke, das viele gute Ideen und Anleitungen enthält. Wie üblich gehe ich in Stufen vor.

```
10 FOR I=0 TO 999
20 POKE 3072+I,42
30 NEXT
```

Diese drei Zeilen zählen vom Anfang des Bildschirmspeichers (3072) 1000 Plätze hoch (von 0 bis 999) und POKEn in jeden Platz einen Stern. Der Bildschirmcode des Sternes ist 42 (siehe oben erwähnte Tabelle).

Auffallend bei diesem eindrucksvollen Vorgang ist, daß der Cursor und die READY-Meldung nicht anschließend beim letzten Stern erscheint, sondern dort, wo der letzte Basic-Befehl – in unserem Fall das RUN – auf den Bildschirm geschrieben worden ist.

Im nächsten Schritt lassen wir den Stern nur auf der obersten und untersten Zeile laufen. Die Schleife zählt daher jetzt nur bis 39 (Zeilenlänge):

```
10 FOR I=0 TO 39
20 POKE 3072+I,42
30 POKE 3072+960+I,42
40 NEXT I
```

Neu ist hier die Zeile 30. Sie erhöht die Adresse um 24x40=960 Plätze und beginnt dadurch in der untersten Zeile. Dieses Programm malt also eine Rahmenlinie oben und unten. Jetzt fehlt noch links und rechts.

```
50 FOR K=0 TO 960 STEP 40
60 POKE 3072+K,42
70 POKE 3072+39+K,42
80 NEXT K
90 GOTO 10
```

Um von oben nach unten zu zählen, beginnen wir mit 0, gehen aber in 40er-Schritten gleich an den Anfang der jeweils nächsten Zeile bis zum Anfang der letzten Zeile. Das gibt uns in Zeile 60 den linken Rand.

	0	1	2	3	4	5	6	7	8	9	10					15					20					25					30					35				39
3072																																								
3112																																								
3152																																								
3192																																								
3232																																								
3272																																								
3312																																								
3352																																								
3392																																								
3432																																								
3472																																								
3512																																								
3552																																								
3592																																								
3632																																								
3672																																								
3712																																								
3752																																								
3792																																								
3832																																								
3872																																								
3912																																								
3952																																								
3992																																								
4032																																								

Bild 4. Aufbau des Bildschirmspeichers

	0	1	2	3	4	5	6	7	8	9	10					15					20					25					30					35				39
2048																																								
2088																																								
2128																																								
2168																																								
2208																																								
2248																																								
2288																																								
2328																																								
2368																																								
2408																																								
2448																																								
2488																																								
2528																																								
2568																																								
2608																																								
2648																																								
2688																																								
2728																																								
2768																																								
2808																																								
2848																																								
2888																																								
2928																																								
2968																																								
3008																																								

Bild 5. Aufbau des Farbspeichers

Für die andere Seite benutzt die Zeile 70 die gleiche Zählung, POKEt aber den Stern um 39 Plätze verschoben, also am rechten Rand. Ganz zum Schluß springen wir in Zeile 90 auf den Anfang zurück, um das lästige READY zu verhindern.

Wenn Sie Schwierigkeiten mit dem Berechnen der Adressen im Bildschirmspeicher haben, dann nehmen Sie bitte Bild 4. Hier sind alle Speicherzellen des Bildschirms mit Zeilen- und Spalten-Nummern angegeben. Diese Darstellung wird Ihnen sicher helfen.

Jetzt wollen wir diese Umrandung bunt machen, und zwar nach jedem Umlauf in einer anderen Farbe. Probieren wir es mit dem COLOR-Befehl in einer übergeordneten Schleife:

```
5 FOR F=1 TO 16
6 COLOR 1,F,4
85 NEXT F
```

Jetzt müßte nach dem ersten Umlauf ein weiterer mit weißen Sternen kommen – tut es aber nicht. Am COLOR-Befehl liegt es nicht, denn wenn Sie ihn mit COLOR 4,F,4 auf die Rahmenfarbe umschalten, zeigt er durchaus sein Können.

Aber: Wenn man in den Bildschirm POKEt, nützt der COLOR-Befehl nichts!

Auch hier bietet der Computer eine Lösung an. Nehmen Sie bitte noch einmal Bild 1 her. Zwischen den Adressen 2048 und 3071 liegt das »Farb-RAM«.

17. Der Farbspeicher

Dieser Bereich im RAM ist der Zwilling des Bildschirmspeicher, ist jedoch nur für die Farben zuständig.

In Bild 5 sehen Sie eine Tabelle, in der die Speicherzelle 2048 genau der Speicherzelle 3072 im Bildschirmspeicher entspricht. Wenn wir jetzt das wiederholen, was wir gleich nach der Überschrift »Bildschirmspeicher« gemacht haben, und POKEn zusätzlich in die 1. Zelle des Farbspeichers 2048 die Zahl 100, dann ändern wir die Farbe des Zeichens, also:

```
- Bildschirm löschen
- Cursor in die untere Hälfte
- POKE 3072,1
- POKE 2048,100
```

Das A erscheint in Lila.

Sie werden jetzt sicher fragen, wie es kommt, daß die Zahl 100 ein Farbwert ist. Die Antwort ist einfach: Erinnern Sie sich, mit dem COLOR-Befehl können wir durch 16 Grundfarben und 8 Helligkeiten 121 verschiedene Farben einstellen. Das können wir im Farbspeicher auch, nur werden da die Farben der Reihe nach numeriert.

Jetzt verrate ich Ihnen noch, daß wir im Farbspeicher 256 Farbwerte verwenden können. Was die Werte über 128 machen, zeigt uns der folgende Einzeiler:

```
10 COLOR 0,1:FOR I=0 TO 255:POKE 3072+I,1:POKE
2048+I,I:NEXT
```

Zuerst setzen wir den Bildschirm (Bereich 0) auf Schwarz (Farbe 1). Bei Schwarz können wir uns eine Helligkeitsangabe sparen. Dann POKEn wir in die ersten 256 Bildschirm-Speicherzellen den Buchstaben A (Bildschirmcode=1). Gleichzeitig POKEn wir in die ersten 256 Farb-Speicherzellen die Farbwerte 0 bis 255.

Nach RUN, das Sie am besten wieder in der Mitte des Bildschirms eintippen, kommen zuerst die 16 Farben in je 8 Helligkeitsstufen. Danach folgen dieselben Farben noch einmal – aber die Buchstaben blinken! Jetzt wissen wir,was passiert, wenn wir die Tasten <FLASH-ON/FLASH-OFF> drücken.

Farbspeicher

1. Im Farbspeicher sind alle Farben gespeichert, in der jedes Zeichen auf dem Bildschirm erscheint.

2. Für den Farbspeicher sind im RAM die Speicherzellen 2048 bis 3071 reserviert.

Da auf dem Bildschirm des C 16 genau 1000 Plätze vorhanden sind (40 Stellen mal 25 Zeilen), reicht der Farbspeicher nur bis Adresse 3048. Die restlichen 24 Byte sind frei.

3. Als Codewerte für die Farben gelten die Zahlen von 0 bis 255. Die Werte 0 bis 15 entsprechen den Farbwerten 1 bis 16 des COLOR-Befehls und zwar in der dunkelsten Helligkeitsstufe 0. Alle nachfolgenden 16er-Blöcke wiederholen die Farben in steigender Helligkeit.

Ab Farbwert 128 wiederholen sich die Farben, nur blinken die Zeiche6n. Die Farbwerte können mit Hilfe der Angaben beim COLOR-Befehl und folgender Formel festgestellt werden:

Speicherfarbe = COLOR-Farbe + Helligkeit x 16 - 1

Wenn Sie die Formel von Punkt 3 der Zusammenfassung anwenden, dann müßten die beiden Befehlskombinationen dasselbe Ergebnis bringen:

(a) COLOR 1,6,4 :PRINT "A"
(b) POKE 3072,1:POKE 2048,69
denn: COLOR-Farbe = 6
Helligkeit = 4
und das gibt: 6 + 4 x 16 - 1 = 69

Wir wollten aber eigentlich das Umrahmungsprogramm mit Farben versehen. Nachdem es mit dem COLOR-Befehl nicht geht, machen wir es mit POKE in den Farbspeicher.

```
5 FOR F=0 TO 255
10  FOR I=0 TO 39
20   POKE 3072+I,170:POKE 2048+I,F
30   POKE 3072+960+I,170:POKE 2048+960+I,F
40  NEXT I
50  FOR K=0 TO 960 STEP 40
60   POKE 3072+K,170:POKE 2048+K,F
70   POKE 3072+39+K,170:POKE 2048+39+K,F
80  NEXT K
85 NEXT F
90 GOTO 5
```

Ich habe folgendes gemacht:

Zeilen 5 und 85 bilden die übergeordnete Schleife, innerhalb der wir die Farbe F in den Farbspeicher POKEn.

Um sicherzustellen, daß die Farbe F in diejenigen Speicherzellen des Farbspeichers kommt, die denen des Bildschirmspeichers entsprechen, habe ich an jeden Bildschirm-POKE-Befehl ein POKE 2048 mit denselben Argumenten angehängt. Nur hinter dem Komma steht ein F für die Farbe und nicht die 1 für den Buchstaben A.

Sie sehen, die Arbeit des Ausrechnens der Adresse braucht man nur einmal zu machen.

Jetzt habe ich noch eine Variante vor:
Die Umrandung soll nicht oben und unten beziehungsweise links und rechts gleichzeitig laufen, sondern immer im Kreis.

Bisher hatten wir diese Situation:
Zeile 20: oben, von links nach rechts
Zeile 30: unten, von links nach rechts
Zeile 60: links, von oben nach unten
Zeile 70: rechts, von oben nach unten.

Wir müssen das so abändern:
Zeile 20: oben, bleibt
Zeile 70: rechts,bleibt
Zeile 30: unten, von rechts nach links
Zeile 60: links, von unten nach oben.

Die Reihenfolge ändert sich also, und die Zeilen 30 und 60 laufen in der entgegengesetzten Richtung. Um das zu erreichen, müssen wir leider für jede POKE-Zeile eine eigene Schleife bauen. In den Zeilen 30 und 60 wird rückwärts gezählt, mit negativem STEP.

In der neuen Reihenfolge und mit 16 Farbwerten im hellen Bereich (96 bis 111) sieht das so aus (Vorsicht, neue Zeilennummern!):

```
5 FOR F=96 TO 111
10  FOR I=0 TO 39
```

```
20   POKE 3072+I,170:POKE 2048+I,F
30  NEXT I
40  FOR K=0 TO 960 STEP 40
50   POKE 3072+39+K,170:POKE 2048+39+K,F
60  NEXT K
70  FOR I=39 TO 0 STEP -1
80   POKE 3072+960+I,170:POKE 2048+960+I,F
90  NEXT I
100  FOR K=960 TO 0 STEP -40
110   POKE 3072+K,170:POKE 2048+K,F
120  NEXT K
135 NEXT F
140 GOTO 5
```

Jetzt setzen wir ganz oben noch ein weiteres Basic-Schmankerl ein – zum Löschen des Bildschirms.

Im Anfangskurs habe ich schon gezeigt, wie man mit dem Befehl PRINT CHR$(147) die Funktion der CLEAR-Taste in ein Programm einbauen kann. Das Basic des C 16 hat hierfür einen eigenen Befehl, er heißt **SCNCLR**, was eine Abkürzung von »Screen Clear« – also »Bildschirm löschen« – ist.

```
3 SCNCLR
```

SCNCLR-Befehl
- dieser Befehl löscht den Bildschirm und gibt READY aus, wenn keine weitere Programmzeile folgt.
- ein im Speicher stehendes Programm wird davon nicht berührt.

Das ganze Umrahmungsprogramm ist in Listing 3 zusammengefaßt:

```
3 SCNCLR
5 FOR F=96 TO 111
10 FOR I=0 TO 39
20  POKE 3072+I,170:POKE 2048+I,F
30 NEXT I
40 FOR K=0 TO 960 STEP 40
50  POKE 3072+39+K,170:POKE 2048+39+K,F
60 NEXT K
70 FOR I=39 TO 0 STEP-1
80  POKE 3072+960+I,170:POKE 2048+960+I,F
90 NEXT I
100 FOR K=960 TO 0 STEP-40
110  POKE 3072+K,170:POKE 2048+K,F
120 NEXT K
130 NEXT F
140 GOTO 5
```

18. Schleifen mit »DO-LOOP«

Schleifen sind Ihnen nichts Neues. Im Listing 3 haben wir nicht weniger als fünf davon eingebaut. Wir haben bisher zwei Methoden des Schleifenbindens gelernt:

```
10 X=X+1                   100 FOR X=0 TO 50
20 PRINT X                 110   PRINT X
30 GET A$                  120   GET A$
40 IF A$="Z" THEN 70       130   IF A$="Z" THEN 150
50 IF X=50 GOTO 99
60 GOTO 10                 140 NEXT X
70 PRINT "STOP"            150 PRINT "STOP"
90 END                     190 END
```

Links steht die Hochzählmethode (Zeile 10) mit der Prüfung (Zeile 50) auf das obere Schleifenende.

Rechts steht die FOR-TO-NEXT-Methode, in der das Hochzählen und die Prüfung auf 50 schon eingebaut ist.

Bei beiden gleich ist die zusätzliche Prüfung mit GET A$, ob während des Ablaufs die Z-Taste gedrückt worden ist, die das Programm frühzeitig abbricht.

Wir müssen übrigens GET und nicht GETKEY nehmen, denn GETKEY wartet auf den Tastendruck, GET aber nicht.

Ich habe außerdem als letzte Zeile den END-Befehl angehängt, damit Sie jedes Programm mit RUN 10 oder RUN 100 einzeln laufen lassen können.

Die FOR-NEXT-Schleife ist wie üblich um eine Zeile kürzer, läuft aber nach Erreichen der 50 über den PRINT-Befehl, während man mit der GOTO-Zählschleife direkt auf das Ende in Zeile 90 springen kann.

Soviel zu den »alten« Methoden. Das Basic des C 16 besitzt zwei weitere Methoden der Schleifenbildung und Abfrage und zwar mit den Befehlen *DO-LOOP* (auf Deutsch »mache – Schleife«) und der Abfrage mit *WHILE* und *UNTIL* (»während«, »bis«). Zusätzlich gibt es noch den Befehl *EXIT* (»Ausgang«).

Obwohl die Namen für sich sprechen, müssen wir uns dennoch die Details ansehen.

Eine ewige Schleife erhalten wir mit DO am Anfang und LOOP am Ende:

```
200 DO
210    PRINT X
220    X=X+1
240 LOOP
```

Um die Schleife nur bis 50 laufen zu lassen, gibt es mit diesen neuen Befehlen fünf Möglichkeiten:

1. mit EXIT

```
300 DO
310    PRINT X
320    IF X=50 THEN EXIT
330    X=X+1
340 LOOP
350 PRINT "STOP"
390 END
```

Die Schleife wird mit den beiden Befehlen DO und WHILE gebildet. Die Prüfung in der Zeile 320 springt mit dem EXIT-Befehl auf die Zeile, die nach dem LOOP-Befehl folgt.

2. mit DO-UNTIL

```
400 DO UNTIL X=51
410    PRINT X
420
430    X=X+1
440 LOOP
450 PRINT "STOP"
490 END
```

Jetzt ist die Prüfung in Zeile 400 eingebaut, allerdings auf 51 als oberste Grenze, Zeile 420 entfällt. Die Schleife läuft solange, bis die UNTIL-Prüfung erfolgreich ist.

3. mit LOOP-UNTIL

```
500 DO
510    PRINT X
530    X=X+1
540 LOOP UNTIL X=51
550 PRINT "STOP"
590 END
```

Diese Schleife hat die UNTIL-Bedingung nach dem LOOP-Befehl; der Effekt ist der gleiche wie bei Nr. 2.

4. mit DO-WHILE

```
600 DO WHILE X <>  51
610    PRINT X
630    X=X+1
640 LOOP
650 PRINT "STOP"
690 END
```

Die Prüfung mit WHILE funktioniert genau umgekehrt wie die UNTIL-Prüfung. Die Schleife läuft solange die Prüfbedingung zutrifft.

5. mit LOOP-WHILE

```
700 DO
710    PRINT X
730    X=X+1
740 LOOP WHILE X <>  51
```

```
750 PRINT "STOP"
790 END
```

Hier steht die WHILE-Prüfung beim LOOP-Befehl, die Wirkung ist dieselbe wie bei Möglichkeit 4.

Der eigentliche Unterschied zwischen DO-LOOP und FOR-NEXT zeigt sich, wenn wir jetzt die zusätzliche GET-Abfrage wie in den Programmen 10-99 und 100-199 einbauen.

Ein Aussprung aus einer FOR-NEXT-Schleife, noch ehe die obere Grenze der Zählvariablen (X) erreicht ist, schließt die Schleife »offiziell« nicht ab, was bei weiteren Rücksprüngen in das Programm zu Fehlern führen kann.

Im Gegensatz dazu ist eine DO-LOOP-Schleife nach einem Aussprung automatisch abgeschlossen und daher gefahrlos.

Ich zeige diesen Aussprung sowohl mit EXIT als auch mit UNTIL:

```
800 DO UNTIL X=51              900 DO UNTIL X=51
810   PRINT X                  910   PRINT X
820   GET A$                   920   GET A$
830   IF A$="Z" THEN EXIT
840   X=X+1                    940   X=X+1
850 LOOP                       950 LOOP UNTIL A$="Z"
860 PRINT "STOP"               960 PRINT "STOP"
890 END                        990 END
```

Interessant ist auch, daß sowohl der DO-Befehl als auch der LOOP-Befehl mit einer Prüfung verknüpft werden kann.

Außerdem ist auffallend, daß DO-LOOP-Schleifen ohne GOTO-Befehle auskommen.

DO-LOOP-Schleifen können genauso verschachtelt werden wie FOR-NEXT-Schleifen. Als Beispiel nehme ich das kleine Programm aus dem Anfangskurs in Sonderheft 5/1986, mit dem ich dort die geschachtelten Schleifen erklärt habe. Dort steht es auf Seite 64 unter dem zugehörigen Bild.

Hier steht dieses Muster rechts, das gleichwertige DO-LOOP-Programm steht links:

```
1000 DO UNTIL X=3          1100 FOR X=1 TO 3
1010  X=X+1                1110   FOR Y=X TO X+2
1020  Y=X
1030   DO UNTIL Y=X+3
1040    PRINT Y            1140   PRINT    Y
1050    Y=Y+1
1060   LOOP                1160   NEXT Y
1070 LOOP                  1170 NEXT X
1090 END                   1190 END
```

Wir sehen, daß die DO-LOOP-Schachtelung aufwendiger ist.

Zusammenfassung
DO-LOOP-EXIT,
DO-LOOP-UNTIL,
DO-LOOP-WHILE

- mit DO beginnt eine Schleife, mit LOOP springt sie auf DO zurück. Alle Programmzeilen zwischen den beiden Befehlen werden als Schleife wiederholt.
- der Befehl EXIT springt aus einer Schleife heraus auf diejenige Programmzeile, die hinter dem LOOP-Befehl folgt. EXIT steht als Ergebnis hinter einer IF-THEN-Prüfung.
- UNTIL kann sowohl als Kombination DO-UNTIL als auch LOOP-UNTIL verwendet werden. Hinter UNTIL folgt eine Prüfungsbedingung, bei deren Eintreten die Schleife beendet wird (Sprung auf Zeile hinter LOOP).
- WHILE ist eine andere Art der Prüfung (hinter DO oder LOOP), welche die Schleife solange laufen läßt, bis die hinter WHILE stehende Prüfbedingung nicht mehr zutrifft.
- DO-LOOP-Schleifen können geschachtelt werden.
- DO-LOOP-Schleifen sind auch bei Ausspringen aus der noch laufenden Schleife abgeschlossen.

Die Vermutung liegt nahe, daß die drei Schleifenprinzipien
- Zählen und Rücksprung mit GOTO
- FOR-TO-NEXT-Schleife
- DO-LOOP-Schleife

sich auch in der Laufzeit unterscheiden.

Sie haben sich wahrscheinlich noch keine Gedanken gemacht, wie lange die Ausführung eines Programms braucht. Aber glauben Sie mir, irgendwann im Lauf Ihrer Programmiertätigkeit werden Sie an einem Punkt ankommen, an dem Ihnen ein Basic-Programm einfach zu langsam ist. Typischerweise passiert das bei Grafik-Programmen.

Bevor Sie dann zu dem Generalbeschleuniger »Maschinensprache« greifen, lohnt es sich durchaus zu überlegen, wie man Basic-Programme beschleunigen kann. Und daß man das kann, habe ich in einer Kursfolge gezeigt, die erst kürzlich in ihrer Gesamtheit wieder veröffentlicht worden ist. Sie finden sie im 64'er-Sonderheft 2/1986 ab Seite 44.

Dabei habe ich eine Methode gezeigt, wie man die Laufzeit eines Programmes messen und auf dem Bildschirm ausdrucken kann. Diese Methode will ich hier verwenden, um die drei Schleifentypen zu testen.

Dazu brauchen wir aber noch zwei weitere Basic-Befehle.

19. Die Uhr des Computers

Der C 16 hat eine innere Uhr eingebaut, deren Stand abgefragt, ausgedruckt und somit zu Messungen und zur Programmsteuerung eingesetzt werden kann.

Diese Uhr startet beim Einschalten des Rechners mit dem Stand 0 und läuft solange, bis sie durch einen entsprechenden Befehl auf Null oder auf irgendeinen anderen Wert gesetzt wird.

Der aktuelle Stand dieser Uhr kann mit dem Befehl *TI* abgefragt werden. Mit der ewigen Schleife:

```
DO:PRINT TI:LOOP
```

drucken wir ein laufendes Band von sich schnell ändernden Zahlen auf den Bildschirm. Diese Zahl durch 60 geteilt gibt uns die Zeit seit dem Loslaufen (Einschalten) in Sekunden an, durch 3600 geteilt in Minuten und durch 216000 geteilt in Stunden.

Weil diese Darstellung etwas mühsam ist, wenn man eine echte Zeitangabe braucht, besitzt Basic noch einen anderen Befehl *TI$*, der die Zeit in einer sechsstelligen Zahl ausdruckt. Dabei bedeutet 124533 12 Stunden, 45 Minuten und 33 Sekunden. Auch diese Abfrage können wir mit einer ewigen Schleife testen:

```
DO:PRINT TI$:LOOP
```

Dieses Zahlenband verändert sich jetzt im Rhythmus von Sekunden.

TI$ ist auch der Befehl, mit dem die Uhr auf einen beliebigen Wert – auch auf Null – gesetzt wird:

```
TI$="000000"
```

setzt die Uhr auf Null.

```
TI$="221500"
```

setzt die Uhr auf 22 Uhr 15.

In dem Augenblick, wo Sie nach diesen Befehlen die <RETURN>-Taste drücken, startet die Uhr mit diesem neuen Anfangswert.

Zusammenfassung Befehle TI, TI$

- Beim Einschalten des Computers startet ein interner Zähler, der 60mal in jeder Sekunde um 1 erhöht wird. Wenn der Zähler den Stand 5184000 erreicht hat – das entspricht einer Laufzeit von 24 Stunden, wird er auf 0 zurückgesetzt.
- Mit TI kann der Stand des Zählers abgefragt werden.
- Der Befehl TI$ wandelt den Zahlenwert des Zählers in eine sechsstellige Zahl um, deren ersten beiden Stellen die Stunden der Uhrzeit, die mittleren beiden die Minuten und die letzten beiden die Sekunden angeben.
- Mit dem Befehl TI$ = "aabbcc" wird die interne Uhr auf aa Uhr bb Minuten.cc Sekunden gestellt.

Jetzt sind wir gerüstet, um die Laufzeit der drei Schleifentypen zu messen.

20. Stoppuhr und Wecker

Unsere Stoppuhr startet in dem Moment, indem sie auf Null gesetzt wird:

```
10 TI="000000"
```

Sie stoppt mit der letzten Zeile der Messung:

```
90 PRINT TI:END
```

Um die Meßzeit in Sekunden zu erhalten, wenden wir noch obige Regel an, teilen durch 60 und drucken aus, was wir sehen werden. Zeile 90 wird verbessert zu:

```
90 PRINT TI/60 "SEKUNDEN"
```

Zwischen 10 und 90 plazieren wir unser Testprogramm. Ich schlage vor, genau wie im schon zitierten Kurs »So macht man Programme schneller« des Sonderheftes 2/1986 den Bildschirm mit 374 A's zu füllen.

Testprogramm »Stoppuhr«

ZÄHLER+GOTO	FOR-TO-NEXT	DO-LOOP
10 TI$="000000"	110 TI$="000000"	210 TI$="000000"
20 SCNCLR	120 SCNCLR	220 SCNCLR
	130 FOR X=0 TO 374	230 DO
40 PRINT "A"	140 PRINT "A"	240 PRINT "A"
50 IF X=374 THEN 80		
60 X=X+1		260 X=X+1
70 GOTO 40	170 NEXT	270 LOOP UNTIL X=375
80 PRINT	180 PRINT	280 PRINT
90 PRINT TI/60"SEK"	190 PRINT TI/60"SEK"	290 PRINT TI/60"SEK"
99 END	199 END	299 END

Die ungleichen Lücken zwischen den Zeilen, sowohl in der Numerierung als auch in der Schreibweise, dienen nur der besseren Lesbarkeit, haben aber auf den Ablauf keinen Einfluß. Die 80er-Zeilen rücken den Ausdruck des Ergebnisses nach unten.

Ergebnis:
ZÄHLER+GOTO-Schleife: 4.4833 SEK.
FOR-TO-NEXT-Schleife: 1.5166 SEK.
DO-LOOP-UNTIL-Schleife: 4.0166 SEK.
Die FOR-NEXT-Schleife ist strahlender Sieger!

Der einzige – für Anfänger sicher nicht gleich erkennbare – Vorteil der DO-LOOP-Schleife bleibt, daß sie auch bei vorzeitigem Abbruch abgeschlossen ist.

In der Überschrift oben habe ich den Wecker erwähnt. Mit TI und TI$ ist er natürlich aktivierbar.

Ein ganz simples Weck-Programm sieht so aus:

```
10 ZEIT$="000020"
20 TI$="000000"
30 DO UNTIL TI$=ZEIT$
40 LOOP
50 VOL 4
60 SOUND 1,600,50
70 SOUND 2,630,50
```

In Zeile 10 wird die Weckzeit eingestellt und der String-Variablen ZEIT$ zugeordnet. Ich habe 20 Sekunden gewählt, damit wir nicht solange warten müssen. In Zeile 20 wird der Wecker gestartet. Zwischen Zeile 30 und 40 läuft eine Schleife, welche prüft, ob der stetig ansteigende Wert der Uhr (TI$) schon der eingestellten ZEIT entspricht. Besser wäre es zu prüfen, ob sie größer ist. Wenn ja, wird unter Lautstärke 4 ein unangenehmer Klang der beiden Tongeneratoren erzeugt.

Wir können übrigens vermeiden, die Weckzeit in Zeile 10 als String eingeben zu müssen. Es gibt nämlich in Basic einen Befehl, der den numerischen Wert eines Strings wiedergibt. Er heißt *VAL*, vom englischen »Value«, was auf Deutsch »Wert« heißt.

Er wandelt einen String in einen Zahlenwert um, sofern der String Zahlen enthält. Diese Zahlen müssen allerdings am Anfang des Strings stehen, wie im folgenden Beispiel gezeigt wird:

```
W$="15 UHR 34":PRINT VAL(W$)
```

Diese Zeile ergibt 15 als Ergebnis. Allzuoft wird dieser Befehl sicher nicht gebraucht. In unserem Fall, wo der Wert von TI$ mit einem anderen Zahlenwert verglichen werden soll, ist er jedoch sehr nützlich.

Das Weckprogramm sieht dann folgendermaßen aus:

```
10 ZEIT=20
20 TI$="000000"
30 DO UNTIL VAL(TI$)=ZEIT
40 LOOP
```

Zusammenfassung VAL-Befehl
- die Schreibweise des Befehls lautet VAL(String). Der String kann sowohl mit dem $-Zeichen als auch innerhalb Gänsefüßen geschrieben sein.
- der VAL-Befehl liefert den numerischen Wert eines Strings. Der String wird dabei Zeichen für Zeichen nach Zahlen abgefragt. Ab dem ersten nicht-numerischen Zeichen stoppt die Suche und der Wert des bisherigen Teils wird weiterverwendet.
- enthält der String keine Zahlen, wird der Wert 0 ausgegeben.

An dieser Stelle muß eigentlich zwangsläufig der Wunsch entstehen, eine Zeitbegrenzung so in ein Programm einzubauen, daß es nach einer gewissen Laufzeit abgebrochen, beziehungsweise beendet wird.

21. Zeitabhängige Programmunterbrechung

Wenn das Programm im Prinzip aus einer oder mehreren Schleifen besteht, ist die Lösung einfach. Ist es ein gerade verlaufendes Programm, das an irgendeiner Stelle unterbrochen werden soll, wird die Sache schon schwieriger, weil die Abfrage der Zeit eigentlich dauernd erfolgen muß.

Wir nehmen hier den leichteren Fall an die Reihe.

Sie kennen doch sicher das alte Spiel »Stadt, Land, Fluß«, bei dem innerhalb einer festgesetzten Zeit aus jedem Sachgebiet ein Beispiel mit denselben Anfangsbuchstaben aufzuschreiben ist. Eine Abwandlung dieses Spiel wähle ich als Anwendung einer Zeitsteuerung, das ich schrittweise mit Ihnen entwickeln möchte.

Mit Initialisierung bezeichnen wir das Herstellen des Anfangszustandes eines Programms. In unserem Fall soll festgelegt werden:
- ein Sachgebiet (S$)
- die Zeitdauer (Z)
- der Anfangsbuchstabe (B$)

Sachgebiet und Zeitdauer werden vom Spieler per INPUT eingegeben, der Buchstabe (von A bis Z) wird mit einem Zufallsgenerator erzeugt.

```
100 SCNCLR
110 INPUT "WAEHLE EIN SACHGEBIET";S$
120 INPUT "STELLE DIE UHR AUF";Z
130 B$=CHR$(INT(RND(0)*26+65))
```

Ich glaube, zu Zeilen 100 bis 120 ist nichts weiter zu sagen.

Zeile 130 soll eine Zufallszahl erzeugen, deren ASCII-Code laut Tabelle auf Seite 215 des Bedienungshandbuches zwischen 65 (A) und 90 (Z) liegt. Das Kochrezept dazu habe ich im Anfangskurs auf Seite 55 des Sonderheftes 5/1986 erklärt. Seine Formel lautet wie folgt:

Ganze Zahlen innerhalb des Zahlenbereiches von minimal Y bis maximal (Y+X-1) werden erzeugt durch:
INT(RND(0)*X+Y)

Zeile 130 wendet diese Formel an. Y liegt mit 65 fest, X errechnet sich aus (Y+X-1)=95 und ergibt 26. Bei Zeile 130 bitte auf die Anzahl der Klammern aufpassen!

Zeile 140 faßt die Initialbedingungen zusammen. Sie ist ein

Beispiel für das Einfügen von Wörtern in einen vorgegebenen Text. Wichtig sind dabei die Leerzeichen im Text vor und nach den Stringvariablen S$ und B$. Versuchen Sie es ohne Leerzeichen, und Sie werden das unlesbare Resultat sehen.

Zeilen 230, 240 und 250 geben den Startschuß.

```
140 PRINT"NENNE "S$", DIE MIT "B$" ANFANGEN"
230 PRINT"START MIT IRGENDEINER TASTE":
240 GETKEY X$
250 TI$="000000"
```

Jetzt startet eine Zeitschleife, die einen vorgegebenen Zeitwert Z mit dem Stand der eingebauten Uhr TI$ vergleicht.

```
260 DO UNTIL VAL(TI$) > Z
340 LOOP
```

Innerhalb der Zeilen 260 und 340 soll folgendes passieren:

- Wörter W$ eingeben
- Anfangsbuchstaben B$ überprüfen
- Wörter ausdrucken
- richtige Wörter zählen (R)

Zum Eingeben von Wörtern stehen uns INPUT und GET zur Verfügung.

INPUT erlaubt die Eingabe von kompletten Wörtern, indem es auf einzelne Zeichen wartet – macht aber damit die Wirkung der Zeitmeßschleife zunichte.

GET wartet nicht und ermöglicht der Zeitschleife, in den Ablauf einzugreifen – nimmt aber nur einzelne Zeichen entgegen.

Die Wirkung der Zeitschleife ist mir wichtiger, zumal man aus einzelnen Zeichen auch Wörter zusammensetzen kann! Wir nehmen also GET (Zeile 270).

Das Zusammensetzen eines Wortes besorgt uns Zeile 280, indem die einzelnen Buchstaben E$ zu einem Wort W$ einfach dazuaddiert werden:

```
270 GET S$
280 W$=W$+S$
```

Mit den vier Zeilen 260, 270, 280 und 340 haben wir eine Schleife, die solange Wörter zusammensetzen würde, bis der Zeitvergleich alles abbricht. Versuchen Sie es einmal mit RUN 260!

Ein Wort wird durch Druck der <RETURN>-Taste abgeschlossen. Wenn das Wort richtig ist, soll es ausgedruckt und als Resultat gedruckt werden. Die primitivste Prüfung ist die des Anfangsbuchstabens.

```
290 IF E$ <> CHR$(13) THEN 340
```

Zeile 290 prüft auf Drücken der <RETURN>-Taste, die den ASCII-Code 13 hat. Das heißt, sie prüft, ob die Taste **nicht** gedrückt ist. Dann springt sie nämlich auf den LOOP-Befehl, springt auf DO zurück, mißt die Zeit und schaut nach, ob ein neuer Buchstabe S$ eingegeben worden ist.

Ist aber <RETURN> gedrückt, geht das Programm mit der Zeile 300 weiter. Diese schneidet sich mit dem LEFT$-Befehl (siehe Anfangskurs, Seite 60) den ersten Buchstaben des Wortes W$ ab und vergleicht ihn mit dem vorgegebenen Anfangsbuchstaben B$.

```
300 IF LEFT$(W$,1) <> B$ THEN 330
310 PRINT W$
320 R=R+1
330 W$=""
```

Ist die Prüfung auf »ungleich« falsch, das heißt, sind beide Buchstaben gleich, dann druckt Zeile 310 das Wort W$ aus und zählt es als richtiges Resultat R zu eventuell schon vorhandenen richtigen Resultaten dazu. Danach wird das eingegebene Wort W$ in Zeile 330 wieder gelöscht.

Ergibt der Vergleich der beiden Buchstaben in Zeile 300 aber, daß sie ungleich sind, dann überspringt Zeile 300 diesen ganzen Teil, löscht in Zeile 330 das bisher eingegebene (aber falsche) Wort und macht in Zeile 340 mit der Schleife weiter.

Nach Ablauf der Schleife, oder besser gesagt, nach Abbruch durch den »Wecker«, läßt Zeile 430/440 die Weckhupe ertönen und Zeile 450 druckt das Resultat R, nämlich die Anzahl der eingegebenen Wörter aus:

```
430 VOL 4
440 SOUND 1,600,50
450 PRINT R
```

In Listing 4a ist das bisherige Programm vollständig dargestellt. Ich habe es dabei mit ein paar REM-Zeilen in funktionelle Blöcke unterteilt.

22. Die Technik der Unterprogramme

Das Spiel »Stadt, Land, Fluß« wird immer mit mehreren Personen gespielt. Obwohl mir klar ist, daß ein derartiges Vorhaben mit dem Computer nicht ganz einfach zu organisieren ist – kein Spieler soll die Wörter des vorhergehenden Spielers sehen können – lasse ich dieses Detail außer acht und zeige Ihnen, wie man so einen mehrfachen Ablauf desselben Programms programmiert.

Wir brauchen dazu folgende Schritte:

- Initialisierung wie vorher
- Frage nach der Zahl der Mitspieler
- Schleife für entsprechend viele Durchgänge
- Einzeldurchgänge pro Spieler
- Speicherung der Einzelergebnisse
- Ausdruck des Gesamtergebnisses

```
100 SCNCLR
110 INPUT "WAEHLE EIN SACHGEBIET";S$
120 INPUT "STELLE DIE UHR AUF";Z
130 B$=CHR$(INT(RND(0)*26+65))
140 PRINT"NENNE "S$", DIE MIT "B$" ANFANGEN"
200 :
210 REM********* ZEITSCHLEIFE & EINGABE *******
220 :
230 PRINT"START MIT IRGENDEINER TASTE":
240 GETKEY X$
250 TI$="000000"
260 DO UNTIL VAL(TI$) > Z
270 GET E$
280 W$=W$+E$
290 IF E$ <> CHR$(13) THEN 340
300 IF LEFT$(W$,1) <> B$ THEN 330
310 PRINT W$
320 R=R+1
330 W$=""
340 LOOP
400 :
410 REM******** STOP & ERGEBNIS ***********
420 :
430 VOL 4
440 SOUND 1,600,50
450 PRINTR
```

Listing 4a

Die Initialisierung mit den Zeilen 100 bis 130 bleibt also gleich.

Jetzt stehen wir vor dem Problem, den eigentlichen Spielteil von Zeile 300 bis Zeile 340 so oft laufen zu lassen, wie Mitspieler vorhanden sind. Völlig unsinnig wäre es, diesen Programmteil zu vervielfachen, da ja die Anzahl der Mitspieler variabel sein soll.

Also müssen wir den Spielteil, der durch die Zeilen 200 bis 340 gebildet wird, bei jedem Spieldurchgang neu »anspringen«. Dazu legen wir ihn zuerst einmal an das Ende des Programms. Das geht am schnellsten, wenn Sie vor jede Zeilennummer eine 1 einfügen, wodurch wir einen Programmblock von 1200 bis 1340 erhalten, die Abstands- und REM-Zeilen mit eingeschlossen.

Vorsicht! Wir haben innerhalb dieser Zeilen zwei Sprung-

adressen, die wir ebenfalls ändern müssen: in Zeile 290 und in Zeile 300.

Und was machen wir mit den »alten Zeilen 200 bis 340 ?

Ein Basic-Befehl, den wir bislang nicht verwendet haben, hilft uns dabei. Er heißt *DELETE*, was »auslöschen« bedeutet. Genau wie beim LIST-Befehl kann man den Bereich angeben, der gelöscht werden soll. In unserem Fall lautet der direkt einzugebende Befehl:

```
DELETE 200-340
```

Wenn Sie das Programm LISTen, dann sitzt jetzt in der Tat der eigentliche Spielteil am Ende des Programms, und Zeilen 200 bis 340 sind verschwunden.

DELETE-Befehl
- der DELETE-Befehl löscht eine oder mehrere Programmzeilen
DELETE 50 löscht nur Zeile 50
DELETE -50 löscht alle Zeilen von 0 bis inklusive Zeile 50
DELETE 50- löscht alle Zeilen ab Zeile 50
DELETE 50-70 löscht von allen Programmzeilen nur die Zeilen 50 bis 70
- der DELETE-Befehl kann nur im Direkt-Modus verwendet werden.

Dasselbe machen wir mit dem anderen Programm-Teilchen »STOP & ERGEBNIS«, das wir ebenfalls öfter verwenden wollen. Wir schieben es mit der oben genannten Methode auf 1400 bis 1450 und DELETEn Zeilen 400 bis 450.

Jetzt haben wir Platz, um die Frage nach der Anzahl der Mitspieler einzubauen. Ich schiebe die Frage noch vor die Spielanweisung in Zeile 140, die damit in Zeile 220 rutscht.

```
140 INPUT "WIEVIELE MITSPIELER";MS
150 SP=1
```

MS ist also die festgelegte Mitspielerzahl, die während eines Spiels konstant bleibt. Da wir aber mit Spieler 1 anfangen und mit Spieler M$ aufhören, brauchen wir noch eine Spieler-Zählvariable, die pro Durchlauf hochgezählt wird. Ich nenne sie »SP« und setze sie in Zeile 150 auf 1.

Als nächstes folgt die große Schleife für die Spieldurchgänge pro Mitspieler. Ihre Zahl ist natürlich von MS abhängig:

```
200 DO WHILE SP <> MS+1
270 SP=SP+1
280 LOOP
```

Die Schleife zwischen den Zeilen 200 und 280 läuft solange, bis SP, welches am Anfang 1 ist und nach jedem Durchgang in Zeile 270 um 1 weitergezählt wird, die volle Mitspielerzahl MS erreicht hat (Prüfung in Zeile 200 auf MS+1). In die Schleife lege ich jetzt die Anweisung der alten Zeile 140 mit der Spielanleitung und der Angabe, welcher Mitspieler an der Reihe ist:

```
210 PRINT"MITSPIELER "SP" IST AN DER REIHE"
220 PRINT"NENNE "CHR$(130) S$ CHR$(132)", DIE MIT
"CHR$(130) B$ CHR$(132) " ANFANGEN"
```

In Zeile 220 habe ich den Ausdruck des Sachgebiets S$ und des Anfangsbuchstabens B$ mit FLASH-ON/FLASH-OFF ausgestattet. Bitte lassen Sie alle Leerzeichen weg, da sonst nicht alles in zwei Zeilen paßt.

Nach Zeile 220 muß jetzt der Sprung auf den Spielteil erfolgen, den wir ans Ende des Programms zwischen die Zeilen 1230 und 1340 geschoben haben.

Bislang sind wir auf solche Stellen mit GOTO 1230 hin- und mit GOTO xxx wieder zurückgesprungen. Basic kennt aber einen Befehl, der das viel eleganter macht, den sogenannten »Unterprogrammsprung«. Er heißt *GOSUB-RETURN*, wobei »Sub« vom englischen »Soubroutine« kommt.

Der Befehl GOSUB steht anstelle des GOTO, RETURN kommt ohne weitere Angaben an den Schluß des anzuspringenden Programmteils, also dorthin, wo wir das zweite GOTO hingesetzt hätten.

In unserem Fall sieht das so aus:

```
230 GOSUB 1230
1350 RETURN
```

Wichtig ist noch zu erwähnen, daß der Programmteil zwischen den Zeilen 1230 und 1350 *UNTERPROGRAMM* heißt.

Wichtig ist außerdem, daß der RETURN-Befehl dann automatisch auf die Zeile nach dem GOSUB springt.

Sinn und Zweck der Unterprogramm-Technik ist es, Programmteile zu bilden, die man nicht nur innerhalb eines Programms mehrfach verwenden kann, sondern die auch in sich beliebig verändert werden können, ohne das Hauptprogramm zu stören. Alles was bekannt beziehungsweise konstant bleiben muß, ist die Anfangs-Zeilennummer des Unterprogramms.

Ein weiterer Vorteil eines Unterprogramms ist, daß es einzeln auf Band oder Diskette in einer »Unterprogramm-Bibliothek« gespeichert und in anderen Programmen eingesetzt werden kann, ohne es immer wieder neu programmieren zu müssen.

Wir erklären den Programmteil »STOP & ERGEBNIS« ab Zeile 1430 ebenfalls zum Unterprogramm und springen nach dem ersten Unterprogramm »ZEITSCHLEIFE & EINGABE« mit einem zweiten GOSUB-Befehl dorthin. Voraussetzung ist, daß es mit RETURN abgeschlossen ist:

```
240 GOSUB 1430
1500 RETURN
```

Wenn Sie das Programm jetzt schon laufen lassen, werden Sie sehen, daß nur noch wenig fehlt.

Zum einen werden die Ergebnisse der einzelnen Mitspieler aufaddiert, zum zweiten fehlt noch eine Rücksetzung vor jedem neuen Spieler, und als letztes brauchen wir noch das Gesamtergebnis.

```
250 SCNCLR
260 R=0
```

Diese beiden Zeilen bewirken das Löschen und die Rücksetzung des Einzelresultats R.

Dieses R müssen wir aber speichern, um es am Ende ausdrücken zu können. Das geschieht am Ende des zweiten Unterprogramms, wo ja in Zeile 1450 das Einzelergebnis R ausgedruckt wird.

Das Gesamtergebnis besteht also aus sovielen Einzelergebnissen, wie Mitspieler vorhanden sind.

Das schreit nach einem Feld!

Da sicher weniger als 11 Mitspieler vorzusehen sind, brauchen wir das Feld nicht zu dimensionieren. Wir teilen die jeweiligen Einzelresultate direkt einer Feldvariablen R(SP) zu, wobei SP ja von 1 bis MS hochgezählt wird. Dadurch werden alle anfallenden Werte von R gespeichert.

```
1450 R(SP)=R
1460 PRINT R(SP)
1470 PRINT"ZUR FORTSETZUNG TASTE DRÜCKEN"
1480 GETKEY A$
```

Im Schlußteil wird das Endergebnis ausgedruckt. Es steht, wie gesagt, in einem Feld R(0), R(1),...R(letzter Spieler), das wir mit einer FOR-NEXT-Schleife ausdrucken:

```
430 FOR I=1 TO (SP-1)
440 PRINT"SPIELER "I" HAT: "R(I)
450 NEXT I
460 END
```

Zeile 460 ist wichtig, da sie das Hauptprogramm von den Unterprogrammen trennt.

Dieses Programm hat uns also in die Technik der Unterprogramme eingeführt, außerdem haben wir durch eine Schleife mit Abfrage der inneren Uhr des C 16 eine Zeitsteuerung des Programms erreicht, wir haben DELETE zur Veränderung von Listings verwendet, haben Töne eingesetzt und haben letztlich wieder einmal eine Feld-Variable verwendet.

In Listing 4b ist das ganze Programm zusammengefaßt.

```
100 SCNCLR
110 INPUT"WAEHLE EIN SACHGEBIET";S$
120 INPUT"STELLE DIE UHR AUF";Z
```

```
130 B$=CHR$(INT(RND(0)*26+65))
140 INPUT"WIEVIELE MITSPIELER";MS
150 SP=1
200 DO WHILE SP<>MS+1
210 PRINT"SPIELER "SP" IST AN DER REIHE"
220 PRINT"NENNE "CHR$(130)S$CHR$(132) " DIE MIT
    " CHR$(130) B$ CHR$(132)" ANFANGEN"
230 GOSUB 1230
240 GOSUB 1430
250 SCNCLR
260 R=0
270 SP=SP+1
280 LOOP
400 :
410 REM ***** ENDERGEBNIS *****
420 :
430 FOR I=1 TO (SP-1)
440 PRINT"SPIELER "I" HAT: "R(I)
450 NEXT I
460 END
1200 :
1210 REM**** ZEITSCHLEIFE & EINGABE ****
1220 :
1230 PRINT"START MIT IRGENDEINER TASTE"
1240 GETKEY X$
1250 TI$="000000"
1260 DO UNTIL VAL(TI$)>Z
1270 GET E$
1280 W$=W$+E$
1290 IF E$<>CHR$(13) THEN 1340
1300 IF LEFT$(W$,1)<>B$ THEN 1330
1310 PRINT W$;
1320 R=R+1
1330 W$=""
1340 LOOP
1350 RETURN
1400 :
1410 REM*** STOP & ERGEBNIS ******
1420 :
1430 VOL 3
1440 SOUND 1,600,50
1450 R(SP)=R
1460 PRINT"ZUR FORTSETZUNG TASTE DRUECKEN"
1470 GETKEY A$
1500 RETURN
```

Listing 4b

Zusammenfassung GOSUB-RETURN-Befehle

- der Befehl wird geschrieben: GOSUB Zeilennummer
- er erzeugt einen Sprung in ein Unterprogramm, welches mit der hinter dem GOSUB stehenden Zeilennummer anfängt.
- das Unterprogramm muß mit RETURN in der letzten Zeile abgeschlossen werden.
- RETURN springt auf die Zeile zurück, die direkt hinter dem GOSUB-Befehl steht.
- Unterprogramme können auch geschachtelt werden
- Unterprogramme können sich bis zu 25 mal selbst aufrufen. Darüber hinaus ist zuwenig Speicherplatz zur Zählung der Aufrufe vorhanden.
- trifft ein Programm auf ein RETURN, ohne vorher ein GOSUB gesehen zu haben, reagiert der Computer mit Abbruch und Fehlermeldung »RETURN WITHOUT GOSUB«

In Listing 4b haben wir die beiden Unterprogramme ans Ende gelegt. Das war durch einfache Umnumerierung der Zeilen leicht zu erreichen. Wir mußten aber ans Ende des Hauptprogramms – noch vor den Unterprogrammen – einen END-Befehl einfügen, um ein unerwünschtes Weiterlaufen des Programms zu verhindern.

Es gibt nun auch die Möglichkeit, Unterprogramme prinzipiell an den Anfang eines Programms zu legen. Das vermeidet den Weiterlauf und hat außerdem eine kürzere Laufzeit. Ich habe das in dem Kurs »So macht man Programme schneller« im Sonderheft 2/1986 auf Seite 49 gemessen und beschrieben.

Ein Programm sieht dann so aus:

GOTO "Hauptprogramm"
Unterprogramm U1
RETURN
Unterprogramm U2
RETURN
"Hauptprogramm"
GOSUB U1
GOSUB U2

Diese Anordnung ist sicher nur dann empfehlenswert, wenn es auf Geschwindigkeit ankommt und viele Unterprogramme verwendet werden.

Programme, die mit vielen Unterprogrammen arbeiten, verwenden oft eine eigene Technik, um »bedienungsfreundlich« zu sein. Mit diesem Begriff bezeichnet die Computerwelt Programme, deren Bedienung so ausgelegt ist, daß man von den Details des Programms oder vom Computer selbst praktisch nichts zu verstehen braucht. Ein wesentlicher Bestandteil davon sind die sogenannten »Menüs«.

23. Bedienungsfreundliche Menü-Technik

Ich schreibe diesen Kurs auf meinem C 64 (den C 16 brauche ich zum Experimentieren) mit einem bekannten Textprogramm, das sich beim Einschalten mit folgendem Menü meldet:

F1 Text bearbeiten
F3 Neuen Text eingeben
F5 Disk-Inhalt
F7 Disk-Befehle
F8 Ende

Je nachdem, welche Funktionstaste ich drücke, kann ich einen bereits angefangenen Text weiterbearbeiten oder mir anschauen, welche Programme sich auf der Diskette befinden oder spät in der Nacht mit F8 Schluß machen!

Jede Überschrift bildet ein eigenes Unterprogramm, auf das durch Drücken der vorgegebenen Funktionstaste gesprungen wird.

Mit unseren heutigen Kenntnissen würden wir diese fünf Funktionstasten mit fünf IF-THEN-Befehlen abfragen. Größere Menüs bräuchten dementsprechend mehr IF-THEN-Zeilen.

In Basic gibt es einen Befehl, mit dem dies sehr viel leichter programmiert werden kann. Er heißt schlicht und einfach »ON« und wird in Verbindung mit dem GOSUB-Befehl verwendet.

Ein ON-GOSUB-Befehl sieht so aus:

ON X GOSUB 100,150,200,250

Diese Zeile ist gleichbedeutend mit:

IF X=1 GOSUB 100
IF X=2 GOSUB 150
IF X=3 GOSUB 200
IF X=4 GOSUB 250

Die Variable X darf also nur die Werte 1, 2, 3, 4 etc. annehmen. Entsprechend springt der GOSUB-Befehl auf die 1., 2., 3. oder 4. Zeilennummer.

Hinter dem ON-Befehl kann auch ein Formelausdruck stehen, nur gilt bezüglich seines Resultats das gleiche wie für die Variable. Es ist klar, daß für Entscheidungen zu Unterprogrammsprüngen nicht immer derartige »saubere« Zahlen zur Verfügung stehen. Damit aber die Umrechnung in die Werte 1, 2, 3 und so weiter nicht aufwendiger als mehrere IF-GOSUB-Befehle wird, werden oft sehr pfiffige Rechen- und Programmierverfahren angewendet. Sie sind es fast immer wert, gesammelt zu werden, wenn man sie in einem Programmlisting antrifft.

Zusammenfassung ON-GOSUB-Befehl

- der Befehl wird folgendermaßen geschrieben:
  ON Variable GOSUB mehrere Zeilennummern
  wobei die Zeilennummern durch Kommata getrennt sein müssen.
- die Variablenwerte legen fest, auf welche der Zeilennummern der GOSUB-Befehl springt und zwar:
  1.0 bis 1.9 1. Zeilennummer hinter dem GOSUB
  2.0 bis 2.9 2. Zeilennummer ...
  3.0 bis 3.9 3. Zeilennummer ...
  und so weiter.
- ist die Variable kleiner 1, springt der GOSUB-Befehl nicht auf eine der Zeilennummern, sondern auf die nach dem ON-GOSUB folgende Zeile.
- ist die Anzahl der Variablenwerte größer als die Anzahl der Zeilennummern, verhält sich der ON-GOSUB-Befehl wie beim Variablenwert 0 oder kleiner 1.
- einen negativen Variablenwert quittiert der Computer mit ILLEGAL QUANTITY.
- hinter dem ON-Befehl kann auch eine Formel oder eine Funktion stehen, nur gilt für ihre Werte dasselbe wie für die Werte einer Variablen.

Zusammenfassung ON-GOTO-Befehl

Für ON-GOTO gilt das gleiche wie für ON-GOSUB, nur werden die Sprünge nach den Regeln des GOTO-Befehls ausgeführt

Ich möchte Ihnen noch schnell die Programmierung des obigen Menüs zeigen.

Ein Menü beginnt immer mit seiner Darstellung auf dem Bildschirm. Das geht einfach über PRINT-Befehle, die nur wenige Zeilen auf dem Bildschirm verteilen müssen. Da es lästig ist, mit programmierten Cursor-Tasten zu operieren, gibt es noch zwei andere Cursorbefehle, die wir gleich anwenden wollen. Der eine heißt *TAB()*, abgeleitet von »Tabulator«, der andere heißt *SPC()*, was vom englischen Wort »Space«, das heißt »Abstand«, kommt. In der Klammer hinter den Befehlen kann eine Zahl von 0 bis maximal 255 stehen.

Beide Befehle werden zusammen mit dem PRINT-Befehl verwendet.

Beim TAB-Befehl rückt der Cursor ausgehend vom linken Rand derjenigen Zeile, auf der er sich gerade befindet, um die in der Klammer stehende Anzahl von Leerstellen weiter.

Der SPC-Befehl tut dasselbe, aber von dem Punkt aus, auf dem sich der Cursor gerade befindet.

```
10 SCNCLR
20 PRINT SPC(90)"WAEHLEN SIE BITTE AUS"
```

SCNCLR löscht bekanntlich nicht nur den Bildschirm, sondern setzt auch den Cursor in die linke obere Ecke. Von dort aus springt er durch SPC(90) 90 Plätze weiter, das sind 80 + 2, also 2 Zeilen und 10 Plätze. Sie können das leicht nach RUN auf dem Bildschirm abzählen.

```
30 PRINT TAB(202)"(1)   TEXT BEARBEITEN"
```

Nach Ausführung der Zeile 20 steht der Cursor in der 3. Zeile. 202 Plätze vom linken Rand der 3. Zeile aus gerechnet sind 5 Zeilen und 2 Plätze; also steht der Cursor in der 9. Zeile am 2. Platz, und die Klammer vor der 1 wird am 3. Platz gedruckt.

Ich will Ihnen gestehen, ich habe es auch nicht gerechnet, sondern einfach ausprobiert. Die nächsten Zeilen verwenden in gleicher Weise den SPC-Befehl.

```
40 PRINT SPC(82)"(2)   NEUEN TEXT EINGEBEN"
50 PRINT SPC(82)"(3)   DISK INHALT"
60 PRINT SPC(82)"(4)   DISK BEFEHLE"
70 PRINT SPC(82)"(5)   ENDE"
```

Die Unterprogramme, die wir über das Menü auswählen wollen, lassen wir ab Zeile 500 beginnen. Ich deute sie natürlich nur an:

```
500 SCNCLR
510 PRINT "TEXT BEARBEITEN"
520 END
530 :
540 SCNCLR
550 PRINT "NEUEN TEXT EINGEBEN"
560 END
570 :
580 SCNCLR
590 PRINT"DISK INHALT"
600 END
610 :
620 SCNCLR
630 PRINT"DISK BEFEHLE"
640 END
650 :
660 SCNCLR
670 PRINT"ENDE"
680 END
```

Die Unterprogramme beginnen also der Reihe nach bei 500, 540, 580, 620 und 660. Dementsprechend lautet der ON-GOSUB-Befehl:

```
110 ON A GOSUB 500, 540, 580, 620, 660
120 PRINT"NULL"
130 END
```

Zeile 120 ist die Ziel-Zeile im Fall, daß für A ein Wert kleiner 1 eingegeben wird.

Die Eingabe erfolgt gewöhnlich mit GETKEY A.

In diesem Fall aber, wo ich Ihnen ermöglichen möchte, auch mit krummen Werten von A zu experimentieren, machen wir es mit INPUT.

```
100 INPUT A
```

So einfach ist das. Bitte experimentieren Sie ein bißchen mit diesem Programm. Komplett steht es in Listing 5.

Zusammenfassung Befehle TAB() und SPC()

- hinter den beiden Befehlen steht eine Zahl von 0 bis 255, die in Klammern stehen muß.
- TAB(I) und SPC(I) werden in Verbindung mit dem PRINT-Befehl verwendet.
- Mit PRINT TAB(I) beginnt der Ausdruck auf dem Bildschirm (oder auf dem Drucker) an dem durch I definierten Platz.
  I=0 definiert den linken Rand der Zeile, auf der sich der Cursor gerade befindet, mit I=20 werden vom linken Rand aus 20 Plätze übersprungen.
- Mit PRINT SPC(I) beginnt der Ausdruck um genau soviele Stellen hinter der augenblicklichen Position des Cursors, wie durch I definiert werden.
  I=0 steht für die augenblickliche Cursorposition, mit I=20 werden 20 Plätze übersprungen.
- beide Befehle springen auf den angegebenen Platz, ohne die übersprungenen Plätze zu löschen.

```
10 SCNCLR
20 PRINT SPC(90)"WAEHLEN SIE BITTE AUS"
30 PRINT TAB(202)"(1)   TEXT BEARBEITEN
40 PRINT SPC(82)"(2)  NEUEN TEXT BEARBEITEN"
50 PRINT SPC(82)"(3)  DISK INHALT"
60 PRINT SPC(82)"(4)  DISK BEFEHLE"
70 PRINT SPC(82)"(5)  ENDE"
80 :
100 INPUT A
110 ON A GOSUB 500,540,580,620,660
120 PRINT"NULL"
130 END
140 :
150 :
500 SCNCLR
510 PRINT"TEXT BEARBEITEN"
520 END
530 :
540 SCNCLR
550 PRINT"NEUEN TEXT EINGEBEN"
560 END
570 :
580 SCNCLR
590 PRINT"DISK INHALT"
600 END
610 :
620 SCNCLR
630 PRINT"DISK BEFEHLE"
640 END 650 :
660 SCNCLR
670 PRINT"ENDE"
680 END
```

Listing 5

Ich habe bei der Aufzählung der Vorteile der Unterprogramm-Technik erwähnt, daß es sinnvoll ist, sich eine Bibliothek von nützlichen Unterprogrammen anzulegen, die dann immer wieder in andere Programme eingebaut werden können.

Nun, das klingt gut und einleuchtend, aber wie hängt man ein Unterprogramm, das in der »Bibliothek« – sprich auf Band oder Diskette – steht, an ein Hauptprogramm im Computer?

Gute Frage, kann ich nur sagen, denn das Basic der Commodore-Computer kennt leider keinen Befehl dafür. Es gibt ihn, bei anderen Computern oder bei Befehlserweiterungsprogrammen. Er heißt *MERGE* oder manchmal auch *APPEND*, was mit »Verschweißen« oder »Anhängen« übersetzt werden kann.

Wir müssen ihn in Basic nachvollziehen, was aber nicht schwer ist.

24. MERGE

Wie gesagt, MERGE ist kein Basic-Befehl, sondern ein Kochrezept.

Ich schreibe Ihnen zuerst das Kochrezept auf und erkläre anschließend, was dabei vorgeht. Es werden folgende Schritte gemacht:

1. Ein Programm steht im Speicher (RAM) des Computers. Ein zweites Programm, das auf Band oder Diskette steht, soll dazugeladen werden.

2. Da das Programm auf Band oder Diskette (2. Programm) hinter das Programm im RAM (1. Programm) gehängt wird, ist es empfehlenswert, daß das 2. Programm höhere Zeilennummern als das 1. Programm hat (eventuell mit RENUMBER herstellen).

3. die folgende Zeile direkt eingeben und mit RETURN abschließen:

```
POKE 43,PEEK(45)-2:POKE 44,PEEK(46)
```

4. Jetzt wird das 2. Programm ganz normal in den Computer geladen (mit LOAD oder DLOAD).

5. die folgende Zeile direkt eingeben und mit RETURN abschließen:

```
POKE 43,1:POKE 44,16
```

Das ist alles; jetzt stehen das 1. Programm und das 2. Programm aneinandergehängt im Speicher des C 16.

Zur Erklärung dieses Kochrezeptes muß ich Sie an Bild 1 erinnern, in dem ich die Speicheraufteilung dargestellt habe. Wie dort gezeigt, beginnt der für Programme verfügbare RAM-Speicher ab Adresse 4096 und geht beim C 16 *ohne* Speichererweiterung bis 16383, beim C 16 *mit* voller Speichererweiterung bis 64767.

In diesem Bereich stehen nun die Zeilen eines Programms, alle Namen und augenblicklichen Werte von Variablen, Felder und schließlich auch alle Texte von String-Variablen. Um ein Chaos zu vermeiden und um die Suchzeiten nach Variablenwerten möglichst kurz zu halten, ist dieser Speicherteil in Bereiche unterteilt.

In Bild 6 ist diese Unterteilung dargestellt.

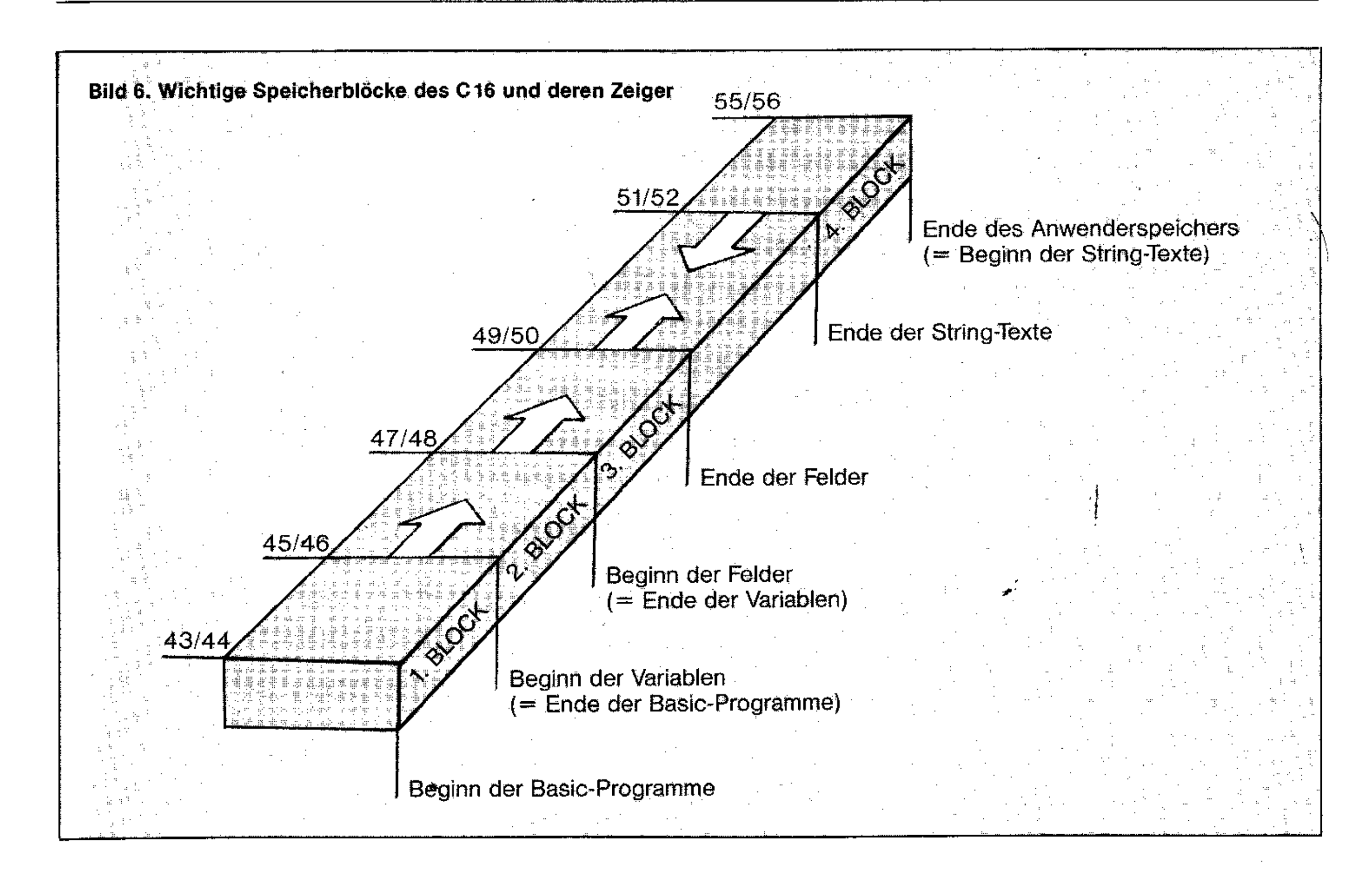


Bild 6. Wichtige Speicherblöcke des C 16 und deren Zeiger

Der Arbeitsspeicher ist, unabhängig von seiner durch Speichererweiterung veränderbaren Größe, in vier Blöcke und einen Freiraum eingeteilt.

- Block 1 beginnt immer an 4096 und enthält die Zeilen (Text) eines eingegebenen Programms. Sein oberes Ende ist variabel, es hängt von der Länge des Programms ab. Deshalb habe ich diese Grenze als Strich mit einem Pfeil dargestellt.
- Block 2 schließt sich direkt an Block 1 an und enthält die numerischen Variablen und die Namen der String-Variablen. Sein Anfang ist identisch mit dem Ende des 1. Blocks und folglich ebenfalls variabel. Dadurch ist auch das Ende von Block 2 variabel. Wenn zum Beispiel in ein Programm weitere Zeilen eingefügt werden, rutschen alle mit Pfeil markierten Grenzen der Speicherbereiche nach oben.
- Block 3 enthält alle Felder.
- Block 4 beginnt am obersten Ende des Arbeitsspeichers und hängt von der Ausbaustufe des Speichers ab.

Er enthält alle Texte der String-Variablen und die der Felder. Je nach Länge beziehungsweise Menge der Texte bewegt sich die Grenze des 4. Blocks solange nach unten bis sie auf die Grenze des 3. Blocks trifft. Dann ist der Speicher voll, und der Computer meldet OUT OF MEMORY.

Jetzt kommt der Aspekt der Speicheraufteilung, der für unser MERGE-Kochrezept wesentlich ist.

Es ist sicher verständich, daß der Computer, oder besser gesagt seine »Betriebsleitung«, jederzeit wissen muß, wo gerade die verschiedenen Grenzen der Blöcke liegen. Das Betriebssystem besitzt dazu im Systemspeicher (von Adresse 0 bis 2047) einige Speicherzellen, in denen der jeweilige Adressenstand der Blockgrenzen gespeichert ist.

Diese Adressen habe ich in Bild 6 am linken Rand eingezeichnet. Sie treten immer paarweise auf. Das liegt daran, daß in einer Speicherzelle allein stets nur Zahlen von 0 bis 255 gespeichert werden können. Das ergäbe 256 Adressen – wir brauchen aber mehr. Wenn man zwei Zellen sozusagen aneinanderhängt, erreicht man dadurch 256 x 256 = 65536 Adressen. Diese doppelte Wortlänge – auch Low-/High-Byte-Darstellung genannt – erfordert natürlich einen Rechenvorgang, um aus dem Inhalt von zwei Speicherzellen die dadurch repräsentierte Adresse zu erhalten.

Wie und warum das so ist, erkläre ich ein paar Absätze weiter unten im Abschnitt »Die Low-/High-Byte-Darstellung«. Hier ist nur die Umrechnungsformel interessant. Steht in den beiden Speicherzellen X und Y eine Adresse, wird diese ausgerechnet mit:

Adresse = X + 256 x Y

In dem Speicherzellenpaar 43/44 steht also die Anfangsadresse des Arbeitsspeichers. Wir prüfen das nach mit:

```
X=PEEK(43):Y=PEEK(44)
PRINT X:PRINT Y
PRINT X+256*Y
```

Das Ganze geht natürlich viel kürzer in einer einzigen Zeile:

```
PRINT PEEK(43) + 256 * PEEK(44)
```

Wir erhalten beim C 16 die Zahl 4097 als Speicheranfang.

Entsprechend steht im Speicherzellenpaar 45/46 die Adresse des Endes von Block 1.

Laden Sie bitte Listing 2 in den Speicher und überprüfen das Blockende mit:

```
PRINT PEEK(45) + 256 * PEEK(46)
```

Bei mir erscheint die Zahl 4235, ich hoffe bei Ihnen auch.

Beim MERGE-Kochrezept führen wir das Betriebssystem des C 16 hinters Licht. Mit den Befehlen:

```
POKE 43,PEEK(45)-2:POKE 44,PEEK(46)
```

schreiben wir in die Speicherzelle 43 diejenige Zahl, um 2 vermindert, die in Speicherzelle 45 steht. Das gleiche machen wir mit der Speicherzelle 44.

Dadurch verschieben wir mutwillig die Anfangsadresse des Arbeitsspeichers an das Ende von Block 1. Da die letzten 2 Speicherzellen hinter einem Programm zur Kennzeichnung des Endes immer nur Nullen enthalten, ziehen wir die 2 ab.

Der Computer glaubt nun, der Arbeitsspeicher beginne bei Adresse 4233 – und ab da lädt er jetzt das Listing 4a, ohne Listing 2 zu berühren.

Danach müssen wir allerdings die Anfangsadresse wie folgt wieder an ihren ursprünglichen Platz schieben:

```
POKE 43,1:POKE 44,16
```

denn 1+256*16 ergibt 4097, was vorher ja dort stand.

Speicherzellenpaare wie 43/44, 45/46 nennen wir *Zeiger*, weil sie auf eine andere Adresse zeigen.

Das Hantieren mit Zeigern ist eine sehr reizvolle Sache. Ich bin sicher, Sie werden noch oft damit konfrontiert. Es gibt viele Aufsätze darüber, ich will mich in diesem Anfängerkurs auf das bisher Gesagte beschränken.

Doch halt, die doppelte Wortlänge muß ich noch erklären.

25. Die Low-/High-Byte-Darstellung

Ich setze voraus, daß Sie das duale Zahlensystem mit seinen Nullen und Einsen kennen, und daß Sie dadurch wissen, was ein Bit ist, außerdem wie man Dualzahlen in Dezimalzahlen umwandelt und umgekehrt.

Sollte dies jedoch nicht der Fall sein, dann lesen Sie bitte im 64'er-Sonderheft 5/1986 auf Seite 90 den Teil Bits und Bytes, der eine kleine Einführung gibt.

Eine Speicherzelle des C 16 hat eine Länge von 8 Bit = 1 Byte.

Mit diesen 8 Bit können Zahlen von 0 bis 255 dargestellt werden. Zur Darstellung von größeren Zahlen verwenden wir die Low-/High-Byte-Methode.

Wir hängen einfach 2 Speicherzellen zusammen, mit deren 16 Bit wir Zahlen bis maximal 65535 darstellen können. Die maximale Zahl 65535 ist übrigens auch die höchste Adresse des gesamten Speichers.

Ich will Ihnen jetzt zeigen, wie eine Dezimalzahl auf zwei 8-Bit-Speicherzellen verteilt wird und umgekehrt: Wie aus zwei Byte eine Dezimalzahl gebildet wird.

Schauen Sie sich das folgende Beispiel an:

Dezimal	47491	
Dualzahl	1011 1001	1000 0011
High-Byte	185	
Low-Byte		131

Wir gehen von der Dezimalzahl 47491 aus. Ihre duale Darstellung mit 16 Bit – 1011100110000011 – teilen wir einfach in der Mitte und erhalten damit zwei neue Dualzahlen mit je 8 Bit = 1 Byte. Das linke Byte nennen wir *High-Byte*, da es den höheren Teil der Gesamtzahl darstellt. Das rechte Byte heißt entsprechend *Low-Byte*.

Jedes der beiden Bytes kann für sich allein in einer Speicherzelle untergebracht werden, in der natürlich dann der dezimale Wert des Bytes steht.

Zur Umrechnung der Low-/High-Bytes empfehle ich folgende Kochrezepte:

Dezimal in Low-/High-Byte

47491:256=185 (High-Byte), Rest 131 (Low-Byte)

Der Rest fällt bei der Division per Hand automatisch an. Mit dem (Taschen)Rechner erhält man den Rest durch:

185 x 256-47491=-131

Low-/High-Byte in Dezimal

High-Byte x 256 + Low-Byte = Dezimal

185 x 256 + 131 = 47491

Eine wichtige Regel gilt es noch zu beachten:

Der Mikroprozessor des C 16 verlangt, daß immer das Low-Byte vor dem High-Byte kommen muß. Die Zahl wird sozusagen von rechts nach links gelesen, in unserem Beispiel 131, 185.

Im Abschnitt »Hat man da Töne?« haben wir ein Programm erstellt, in dem Töne als Variablenwerte in ein Feld gespeichert wurden. Wir waren damals noch nicht in der Lage, diese Töne auf Band oder Diskette abzuspeichern. Ich habe Sie damals gebeten, das Programm-Fragment abzuspeichern, bis wir soweit wären, mehr vom Speicher und vom Speichern zu wissen.

Jetzt sind wir soweit. Ein Blick auf Bild 6 macht es deutlich.

Mit den Befehlen LOAD beziehungsweise DLOAD speichern wir nur den Inhalt des 1. Blocks ab. Die Töne stecken aber im 3. Block.

26. Der Unterschied zwischen Programm und Datei (File)

Der Computer unterscheidet glücklicherweise beim Speichern auf externe Speichergeräte (Datasette, Diskette) zwischen Daten aus dem 1. Block und den Blöcken 2, 3 und 4. Daten aus den letzteren nennen wir eine *Datei* oder auf englisch *File*.

Ein *Programm* besteht aus einer Ansammlung von numerierten Zeilen. Jede Zeile enthält Anweisungen an den Computer. Diese Anweisungen werden nach RUN zeilenweise in Reihenfolge der aufsteigenden Zeilennummern ausgeführt.

Eine *Datei* (File) besteht aus rohen Daten, ohne jede Angabe, was damit gemacht werden soll.

Der bekannte Computer-Journalist Richard Mansfield hat zur Erklärung dieses Unterschieds einmal das einleuchtende Beispiel einer Dose mit Tomatensuppe gebracht. Der Aufkleber der Dose enthält sowohl eine Datei als auch ein Programm.

Die Datei lautet:
Wasser, Tomaten, Salz,
Monosodiumglutamat, Farbstoff,
Oleoresin

Das Programm lautet:

1. Vorsichtig öffnen
2. Inhalt in einen Topf leeren
3. eine Tasse Wasser hinzufügen
4. auf kleiner Flamme erhitzen
5. ungewürzt servieren

Ich bin sicher, Sie sehen jetzt den Unterschied.

Programme werden mit SAVE auf Band und mit DSAVE auf Diskette gespeichert.

Dateien werden mit einer eigenen Befehlskombination OPEN, PRINT# und CLOSE sowohl auf Band als auch auf Disketten gespeichert.

Genau das wollen wir jetzt lernen.

27. Dateien (Files) speichern und laden

Als erstes will ich darauf hinweisen, daß über Datei mit dem C 16 im 64'er-Sonderheft 3/1986, ein ausgezeichneter Aufsatz von Said Baloui, »Daten verwalten mit Datasette« erschienen ist. Herrn Balouis Konzentrierung auf die Arbeit mit dem Band nehme ich zum Anlaß, hier mehr über Disketten-Dateien zu schreiben.

Wenn man eine Datei laden oder speichern will, muß zu allererst eine Verbindung zwischen Computer und dem externen Speichergerät hergestellt werden. Ich meine damit nicht das Verbindungskabel, sondern eine Anweisung an den Computer, sich für den Transfer von Daten über das externe Gerät bereitzuhalten.

Man nennt das auch:
»eine Datei eröffnen«.

Es ist vergleichbar mit dem Herausziehen und Öffnen eines Karteikastens, um Zugriff zu den Daten auf den Karteikarten zu haben.

Der Befehl dazu heißt *OPEN*, was natürlich »Öffnen« heißt.

OPEN-Befehl
- der OPEN-Befehl öffnet eine Datei zum Schreiben oder Lesen. Mit diesem Befehl wird automatisch ein Pufferspeicher angelegt, mit dem die verschiedenen Arbeitsgeschwindigkeiten der externen Speichergeräte ausgeglichen werden können.
- hinter dem OPEN-Befehl stehen bis zu 6 weitere Angaben, die durch Kommata voneinander getrennt sein müssen. Diese Angaben sind: OPEN Dateinummer,Gerätenummer, Sekundäradresse,»Dateiname,TYP,Modus«

Die zusätzlichen Angaben hinter OPEN bedürfen einer Erklärung. Sie sind sicher für den Anfänger verwirrend, und wir brauchen sie jetzt auch gar nicht alle. Man findet jedoch selten eine komplette Übersicht, daher mache ich die Aufstellung vollzählig – für später.

Dateinummer
ist die Kennzeichnung der eröffneten Datei. Sie kann eine Zahl zwischen 1 und 255 sein. Sie ist wichtig und dient als Referenz beim Umgang mit dieser bestimmten Datei; dies umso mehr, als gleichzeitig bis zu 10 verschiedene Dateien geöffnet sein können.

Gerätenummer
ist eine Kennzahl dafür, mit welchem externen Gerät die Verbindung hergestellt wird. Dabei gilt:
0 Tastatur
1 Kassettenrecorder (Datasette)
3 Bildschirm
4 Drucker 1
5 Drucker 2
8 Diskettenlaufwerk 0
9 Diskettenlaufwerk 1
4 bis 127 Geräte, die am seriellen Bus angeschlossen sind
128 bis 255 Geräte über seriellen Bus, mit einem Zeilenvorschub nach dem Zeilenende.
Übrigens: Wenn die Gerätenummer weggelassen wird, stellt der Computer automatisch die 1 für Datasette ein.

Sekundäradresse
kann eine Zahl zwischen 0 und 255 sein. Diese Zahl hat bei den verschiedenen externen Geräten unterschiedliche Bedeutung.

Tastatur		keine Wirkung
Kassettenrecorder	0	vom Band lesen
	1	auf Band schreiben
	2	auf Band mit »Bandende«-Markierung schreiben
Bildschirm		keine Wirkung
Drucker	0 bis 10	die Funktionen sind bei den Druckerfabrikaten zum Teil unterschiedlich, bitte im Drucker-Handbuch nachsehen
Diskettenlaufw.	0 u. 1	vom Betriebssystem reserviert
	2 bis 14	reserviert einen numerierten Datenkanal, bis zu drei gleichzeitig
	15	reserviert einen Befehlskanal

Dateiname
steht immer zwischen Anführungszeichen. Er kann bei der Datasette auch weggelassen werden.

Dateityp
wird nur bei Diskettenoperationen verwendet und steht innerhalb der Anführungszeichen hinter dem Dateinamen; er bezeichnet die folgenden Typen:
S sequentielle Datei
U User-Datei
P Programm-Datei
R relative Datei
Wir werden uns im Rahmen dieses Kurses nur mit sequentiellen Dateien befassen, erstens weil mit der Datasette nur dieser Typ möglich ist und zweitens, weil er leichter zu verstehen ist. Den Unterschied zwischen den Dateitypen hat Herr Baloui im oben zitierten Aufsatz beschrieben.

Modus
wird nur bei Diskettenoperationen verwendet und legt fest, wie der Datenkanal genutzt werden soll:
W schreiben einer Datei (Write)
R lesen einer Datei (Read)
A verlängern einer sequentiellen Datei (Append)
M lesen einer nicht geschlossenen Datei
Nun bitte keine Panik!
Ich habe gesagt, daß ich hier nur eine vollständige Nachschlageliste anlege. In diesem Kurs brauchen wir nur Bruchteile daraus. Aber immerhin ist das alles ein guter Fingerzeig, daß speziell Diskettenoperationen sehr raffiniert sein können.
Zurück zur Anwendung des OPEN-Befehls. Mit:

```
10 OPEN 1,1,1,"TEXT"
```

eröffnen wir also eine Datei namens »TEXT« auf der Datasette.
Für die Diskette lautet derselbe Befehl:

```
10 OPEN 1,8,3,"TEXT,S,W"
```

»S« steht für sequentielle Datei, »W« für schreiben.
Um jetzt Daten auf das externe Gerät schreiben zu können, verwenden wir eine Variante des PRINT-Befehls, der **PRINT#** heißt. Hinter PRINT# muß immer die Dateinummer stehen, die beim OPEN-Befehl verwendet worden ist. Danach folgt die zu schreibende Variable.

```
20 PRINT#1,"FUSSBALL"
30 PRINT#1,"TENNIS"
```

Damit haben wir die zwei Wörter abgesendet. Hinter einem PRINT#-Befehl darf immer nur eine Variable oder ein String stehen, damit die Daten in der Datei einen Abstand haben. Als nächstes muß die eröffnete Datei wieder geschlossen werden, der entsprechende Schließbefehl lautet *CLOSE*. Er wird mit der Nummer der Datei versehen.

```
40 CLOSE 1
50 END
```

Wenn Sie diese vier Zeilen laufen lassen, fordert die Datasette zum Drücken der RECORD- und PLAY-Taste auf, das Diskettenlaufwerk startet sofort und die Datei ist gespeichert.

PRINT#-Befehl
- der PRINT#-Befehl sendet Daten einer Datei auf Band oder Diskette. Maximal können mit einem PRINT#-Befehl 255 Zeichen gesendet werden
- dem PRINT#-Befehl muß dieselbe Dateinummer folgen, wie dem OPEN-Befehl, der die Datei eröffnet hat
- es können numerische und String-Variable sowie Feld-Variablen gesendet werden. Wichtig ist, daß die einzelnen Variablen durch das RETURN-Zeichen voneinander getrennt sind. RETURN kann entweder durch separate PRINT#-Befehle oder durch CHR$(13) erzeugt werden

CLOSE-Befehl
- dieser Befehl schließt eine eröffnete Datei
- der CLOSE-Befehl wird lediglich mit der Nummer der Datei versehen, die er schließen soll
- jede Datei muß unbedingt geschlossen werden, da andernfalls keine Endmarkierung hinter die letzte Dateneintragung geschrieben wird. Dadurch können die letzten Daten verloren gehen. Im Inhaltsverzeichnis (Directory) einer Diskette ist eine nicht geschlossene Datei mit einem Stern * gekennzeichnet.

Jetzt kommt der andere Teil – eine auf Band oder Diskette gespeicherte Datei in den Computer laden oder lesen. Die Datei muß wieder eröffnet werden, diesmal zum Lesen.
Für die Datasette lautet der Befehl:

```
100 OPEN 1,1,0,"DATEI"
```

Für die Diskette dagegen muß es heißen:

```
100 OPEN 1,8,3,"DATEI,S,R"
```

Um Daten einer Datei zu lesen, gibt es gleich zwei Befehle, die Sie in ähnlicher Form schon kennen, nämlich *INPUT#* und *GET#*.

GET#-Befehl
- Der GET#-Befehl funktioniert genauso wie der normale

GET-Befehl: Er liest einzelne Zeichen einer geöffneten Datei von einem Eingabegerät. Die Anzahl der Zeichen (Länge der Strings) ist beliebig.
- Hinter GET# muß die Datei-Nummer des OPEN-Befehls stehen.
- GET# kann nicht im Direkt-Modus verwendet werden.

INPUT-Befehl
- Der INPUT#-Befehl liest, wie GET#, Daten einer mit OPEN eröffneten Datei externer Eingabegeräte.
- Hinter INPUT# muß die Dateinummer des OPEN-Befehls stehen.
- Während GET# Zeichen für Zeichen liest, holt INPUT# ganze Datengruppen.
- Hinter einem INPUT#-Befehl können mehrere, durch Kommata getrennte Variablen stehen.
- Ein String darf nicht länger als 80 Zeichen sein.
- INPUT# kann nicht im Direkt-Modus verwendet werden.

In unserer Datei haben wir zwei Strings gespeichert. Mit folgender Zeile lesen wir sie:

```
110 INPUT#1,A$,B$
120 PRINT A$,B$
130 CLOSE 1
```

In Zeile 120 trenne ich das Ausdrucken von A$ und B$ mit einem Komma. Mit dem Klebeeffekt des Semikolon würde nämlich das Wort FUSSBALLTENNIS ausgedruckt werden. Probieren Sie das kleine Leseprogramm mit RUN 100 aus.

Das ganze Listing für Schreiben und Lesen einer *String-Datei* sieht so aus:

```
10 OPEN 1,1,1,"TEXT"          ...für Band
10 OPEN 1,8,3,"TEXT,S,W"      ...für Diskette
20 PRINT#1,"FUSSBALL"
30 PRINT#1,"TENNIS"
40 CLOSE 1
50 END
60 :
100 OPEN 1,1,0,"DATEI"        ...für Band
100 OPEN 1,8,3,"DATEI,S,R"    ...für Diskette
110 INPUT#1,A$,B$
120 PRINT A$,B$
130 CLOSE 1
```

Für eine *Datei mit numerischen Variablen* sieht das Programm fast gleich aus:

```
200 OPEN 1,1,1,"ZAHLEN"       ...für Band
200 OPEN 1,8,3,"ZAHLEN,S,W"    ...für Diskette
210 PRINT#1,35
220 PRINT#1,14
230 PRINT#1,753
240 PRINT#1,2
250 CLOSE 1
260 END
270 :
300 OPEN 1,1,0,"Zahlen"       ...für Band
300 OPEN 1,8,3,"ZAHLEN,S,R"   ...für Diskette
310 INPUT#1,A,B,C,D
320 PRINT A;B;C;D
330 CLOSE 1
```

Bei Zahlen dürfen wir in Zeile 320 ruhig Semikolons verwenden, da Zahlen automatisch Trennungsabstände haben.

Wir wollen noch die Wirkungsweise des GET#-Befehls ausprobieren. Ersetzen Sie im ersten Programm in Zeile 110 das INPUT# durch GET#:

```
110 GET#1,A$,B$
```

Nach RUN 100 erhalten wir den weit auseinandergezogenen Ausdruck der Buchstaben F und U. So geht es also nicht – GET# nimmt immer nur ein Zeichen. Wir müssen GET# am besten mit einer Schleife wiederholt einsetzen.

```
110 GET#1,A$
120 PRINT A$;
125 GOTO 110
```

Jetzt erhalten wir die beiden Wörter FUSSBALL und TENNIS, aber die Schleife stoppt nicht. Wir prüfen daher, wann das letzte Zeichen erscheint, und CLOSEn dann die Datei. Dazu müssen wir im Eingabeteil (Zeile 30) hinter dem letzten String ein erkennbares Zeichen einsetzen. Ich habe einfach hinter Tennis ein Leerzeichen gesetzt.

Leider müssen wir dadurch die Datei neu eingeben, am besten mit einem anderen Namen in Zeile 10 und 100. Oder aber Sie löschen die alte Datei »TEXT«.

```
30 PRINT#1,"TENNIS "
123 IF A$=" " THEN 130
```

Das sind die geänderten Zeilen.

Das wiederholte Schreiben der einzelnen PRINT#-Befehle ist natürlich sehr lästig. Besser geht das mit einer FOR-NEXT-Schleife, wobei die zu speichernden Daten entweder direkt per normalen INPUT-Befehl eingegeben oder mit READ aus gespeicherten DATA-Zeilen geholt werden.

Ich zeige das an unserer TEXT-Datei in der GET-Version:

```
10 OPEN 1,8,3,"TEXT,S,W"      ...für Diskette
10 OPEN 1,1,1,"TEXT"          ...für Band
20 FOR I=1 TO 3
30 READ A$
40 PRINT#1,A$
50 NEXT I
60 CLOSE 1
70 DATA FUSSBALL,TENNIS,*
80 END
90 :
100 OPEN 1,8,3,"TEXT,S,R"     ...für Diskette
100 OPEN 1,1,0,"TEXT"         ...für Band
110 GET#1,A$
120 IF A$="*" THEN 130
123 PRINT A$;
125 GOTO 110
130 CLOSE 1
```

Ich habe die Zeile 120, die auf Datei-Ende prüft, leicht geändert. Sie prüft jetzt auf ein abschließendes Zeichen »*«, das ich als letzte Eintragung in die neue DATA-Zeile 70 geschrieben habe. READ und PRINT#1 erfolgen nun innerhalb einer Schleife zwischen Zeile 20 und 50.

Sie müssen nun die Datei TEXT entweder löschen (Diskette) oder überschreiben (Band). Mit RUN läuft der Schreibteil des Programms bis Zeile 80. Mit RUN 100 starten Sie den Leseteil.

Zum Schluß meiner Datei-Betrachtungen will ich mein Versprechen einlösen, das ich bei den zweidimensionalen Feldern gemacht habe und Ihnen noch zeigen, wie wir die Melodie, die wir »komponiert« und in Felder gelegt haben, abspeichern und später variieren können.

Wir können auch eine *Datei mit Feldern* anlegen.

Holen Sie bitte das »Fragment« des Melodie-Programms in den Computer. Es lautet:

```
10 DIM M(17,2)
20 FOR I=0 TO 17
30 INPUT "STIMME,TON,DAUER";S,T,D
35 M(I,0)=S
40 M(I,1)=T
50 M(I,2)=D
60 NEXT I
70 FOR I=0 TO 17
80 VOL 3
90 SOUND M(I,0),M(I,1),M(I,2)
100 NEXT I
110 END
```

Mit diesem Teil werden 18 Töne in das Feld M (17,2) gebracht und auch einmal laut gespielt.

Der nächste (neue) Programmteil speichert die Töne in einer Datei namens »MELODIE« ab.

```
120 :
130 REM*** TOENE SPEICHERN *****
140 :
200 OPEN 1,8,3,"MELODIE,S,W"    ...für Diskette
200 OPEN 1,1,1,"MELODIE"        ...für Band
210 FOR I=0 TO 17
220 PRINT#1,M(I,0)
230 PRINT#1,M(I,1)
240 PRINT#1,M(I,2)
250 NEXT I
260 CLOSE 1
270 END
```

Dieser Schreibteil des Programms sieht auch nicht anders aus als die Dateien; die Feld-Variablen werden wie andere Variablen behandelt.

Der Leseteil allerdings muß mit der Dimensionierung eines Feldes beginnen, denn da hinein bringen wir ja die aus der Datei gelesenen Töne.

```
280 :
290 REM***** TOENE LESEN & SPIELEN *****
295 :
300 DIM M(17,2)
310 OPEN 1,8,3,"MELODIE,S,R"    ...für Diskette
310 OPEN 1,1,0,"MELODIE"        ...für Band
320 FOR I=0 TO 17
330 INPUT#1,M(I,0),M(I,1),M(I,2)
340 NEXT I
350 CLOSE 1
360 GOTO 70
```

Auch hier brauche ich nichts erklären. Nach RUN 300 werden die Töne gelesen. Der Sprung auf Zeile 70 des Erzeuger-Programms, falls es noch im Speicher des Computers steht, läßt die Melodie erklingen. Sie können den VOL-SOUND-Teil natürlich auch in die Schleife zwischen Zeile 320 und 340 einbauen. Die Tondauer D=60 ist sicher länger als die Lesezeit der Datei.

28. Die Funktionstasten des C16

Im Anfangskurs (Sonderheft 5/1986) habe ich auf Seite 61/62 die Abfrage der Funktionstasten beschrieben.

Da beim C16 die Funktionstasten aber bereits mit Funktionen belegt sind, geht das so nicht. Wir erhalten noch nicht einmal die ASCII-Codes der Funktionstasten, wenn wir mit der folgenden Schleife die Tasten abfragen:

```
10 DO
20 GETKEY A$
30 A=ASC(A$)
40 PRINT A
50 LOOP
```

Nach RUN erhalten wir die ASCII-Codewerte aller Tasten, auch die der Steuertasten, nur nicht der Funktionstasten. Diese liefern uns nämlich die ASCII-Codewerte der Buchstaben ihrer fest eingestellten Funktionen.

Mit dem Befehl *KEY* können wir diese Funktionen ändern. Seine Übersetzung bedeutet »Schlüssel«.

KEY allein eingegeben druckt die derzeitigen Funktionen der Tasten aus. Mit KEY 5,»Funktion« wird der F5-Taste die in Anführungszeichen stehende Funktion zugeordnet.

```
KEY 1,"GET A$
```

Mit dieser Zeile, direkt eingegeben, ist die F1-Taste mit unserer alten GET-Schleife (die wir durch GETKEY ersetzt haben) belegt.

Ähnlich unpraktisch wäre die folgende Belegung der F3-Taste:

```
KEY 3,"HINTERHUBER"
```

Aber so jedenfalls funktioniert das Belegen der Funktionstasten mit eigenen Funktionen oder Befehlen.

Mit derselben Methode kann man aber die Funktionstasten »löschen«, so daß sie wie beim C64 oder VC 20 als Steuertasten abfragbar sind. Wir belegen sie ganz einfach mit ihren üblichen ASCII-Codewerten:

```
KEY1,CHR$(133)
KEY2,CHR$(137)
```

Die vollständige Liste ist in Tabelle 1 enthalten.

Wenn Sie jetzt das kleine Abfrageprogramm von oben starten, erscheinen auch die ASCII-Codewerte der Funktionstasten. Damit sind dann die Abfrageprogramme des Anfangskurses auf dem C16 funktionsfähig.

Wenn aber der Computer aus- und wieder eingeschaltet, oder die Reset-Taste gedrückt wird, sind die fest eingestellten Funktionen wieder da.

Zusammenfassung KEY-Befehl
- mit dem KEY-Befehl können die vordefinierten Funktionstasten mit anderen Funktionen der Befehlen belegt werden.
- die Schreibweise dieses direkt eingebbaren Befehls ist: KEY Tastennummer,»Funktion«
- wird der Befehl KEY ohne weitere Angaben eingegeben, druckt er die derzeit vorgegebenen Funktionen der Tasten aus.

29. Zweiteilige Schleifen

Wir kennen bisher den IF-THEN-Befehl als Methode, um eine Entscheidung innerhalb eines Programms herbeizuführen.

Dabei gilt, daß bei Erfüllung der Prüfbedingung IF alles ausgeführt wird, was hinter dem THEN steht, auch die Sprungbefehle. Ist die Bedingung nicht erfüllt, macht das Programm in der nächsten Zeile weiter. Als Beispiel dafür dient das folgende kleine »Würfel«-Programm, das statistisch auswertet, wieviele der gewürfelten Zahlen (von 1 bis 6) gerade und wieviele ungerade sind.

```
10 DO
20 Z = INT(RND(0)*6+1)
30 IF INT(Z/2)=Z/2 THEN G=G+1:GOTO50
40 U=U+1
50 PRINT G,U
60 LOOP
```

In Zeile 20 wird nach der bereits bekannten und des öfteren eingesetzten Formel eine Zufallszahl Z zwischen 1 und 6 erzeugt.

Der Ausdruck IF INT(Z/2)=Z/2 prüft, ob die Zahl Z gerade ist. Zum Beispiel ist Z=3, dann ist Z/2 = 1,5 und INT(Z/2) = 1, so daß der Vergleich nicht richtig ist. Er ist nur bei geraden Zahlen richtig.

Bei geraden Zahlen wird noch in Zeile 30 die Summe der geraden Zahlen G um 1 erhöht, und das Programm springt dann auf Zeile 50 zum Ausdruck der Resultate weiter.

Ist die Zahl aber ungerade, dann macht das Programm in der Zeile nach dem IF-Befehl weiter. Dort, in Zeile 40, wird dann die Summe der ungeraden Zahlen U um 1 erhöht.

Das Basic des C16 besitzt eine Erweiterung des IF-THEN-Befehls, mit dem die Entscheidung in den Zeilen 30 und 40 verbessert werden kann. Der Befehl heißt *IF-THEN-ELSE*.

Ist hier die Prüfbedingung erfüllt, wird alles, was hinter THEN steht, ausgeführt. Ist die Bedingung aber nicht erfüllt, geht das Programm mit dem ELSE-Teil weiter.

Das »Würfel«-Programm sieht jetzt so aus:

```
100 DO
110 Z = INT(RND(0)*6+1)
120 IF INT(Z/2)=Z/2 THEN G=G+1:ELSE U=U+1
130 PRINT G,U
140 LOOP
```

Zwischen THEN und ELSE beziehungsweise nach dem ELSE können noch weitere Befehle folgen. Sie zählen jeweils zum THEN- oder zum ELSE-Teil.

Als Beispiel dafür drucken wir zusätzlich die Zahl Z aus. Die verschiedenen Kommata sorgen für die entsprechende Verteilung der Zahlen auf dem Bildschirm. PRINT,,Z druckt bekanntlich die Zahl Z ab der Hälfte des Bildschirms, weil das Komma beim PRINT-Befehl den Bildschirm in Viertel teilt.

```
120 IF INT(Z/2)=Z/2 THEN G=G+1:PRINT Z:ELSE U=U+1:
    PRINT,Z
130 PRINT,,G;U
```

Es bleibt neben der Programmiererfahrung, die Sie durch dieses Beispiel gewonnen haben, nur noch Ihrem Spieltrieb überlassen, diese Kleinstatistik auszuwerten oder für andere Zahlenmengen umzuwandeln.

Auch die Zufälligkeit von Zufallszahlen läßt sich hier erproben, insbesondere, wenn Sie die verschiedenen, im Anfangskurs auf Seite 54 oder in der 64'er-Ausgabe auf Seite 131 beschriebenen verschiedenen Methoden der Zufallszahlenerzeugung anwenden.

Zusammenfassung IF-THEN-ELSE-Befehl
- Wenn die Bedingung nach dem IF erfüllt ist, werden die auf das THEN folgenden Befehle ausgeführt.
- Ist die Bedingung hinter dem IF nicht erfüllt, werden die auf das ELSE folgenden Befehle ausgeführt.

30. Zusammenfassung

Wie schon im Anfangskurs im Sonderheft 5/1986 ist es mir auch hier nicht gelungen, alle Basic-Befehle zu behandeln. Meine Erklärungen lassen einfach eine kurze gedrängte Zusammenstellung nicht zu.

Aber ich wollte ja auch nicht ein zweites Betriebshandbuch schreiben, sondern Ihnen an vielen Beispielen, Tips und Kochrezepten die »Kunst« des Programmierens näherbringen.

Ich verweise Sie nochmals auf die anderen, hier immer wieder zitierten 64'er-Sonderhefte und auf das 64'er-Stammheft. Das ganze Thema der Grafik zum Beispiel ist in mehreren Beiträgen im Sonderheft 5/1986 behandelt. Weitere Beiträge werden sicher noch folgen.

Besonders im Stammheft werden regelmäßig Aufsätze und Tips sowohl für die Programmierung als auch über die Arbeitsweise des C16 veröffentlicht.

Mir bleibt jetzt nur noch die Aufgabe, alle diejenigen Befehle aufzulisten, die ich nicht behandelt habe.

Programmierhilfen:
Die Funktionen von
AUTO, CONT, TROFF, TRON, DS, DS$
sind im Handbuch von Commodore recht gut beschrieben.

Fehlerbehandlung:
Die Befehle
RESUME, TRAP, EL, ER, ERR$ und ST
sind sicher eine eigene Behandlung wert.

Stringbearbeitung:
Hier fehlen nur
INSTR, STR, und die zusätzlichen Eigenschaften von MID$

Grafikbefehle:
Dazu gehören
BOX, CHAR, CIRCLE, DRAW, GRAPHIC, PAINT, LOCATE, RDOT, RGR, SCALE und alle SHAPE-Befehle.
Wie schon gesagt, auch das ist ein Kurs für sich.

Diskettenbefehle:
Die meisten sind im Handbuch beschrieben. Diejenigen, die in das Innenleben der Disketten führen, müssen in einem Sonderkurs behandelt werden.

Numerische und trigonometrische Funktionen:
Um diese tut es mir leid, denn sie kommen immer wieder zu kurz. Auch hier verweise ich auf das Handbuch von Commodore.

Programm- und Systembefehle:
Hier fehlen nur noch
PRINT-USING, WAIT, DEF-FN, SYS und USR.

DEF und WAIT sind für Anfänger durchaus verwendbar, die anderen drei gehören eigentlich zur fortgeschrittenen Programmierung.

Ich hoffe, Sie haben Spaß an dem Kurs. Wenn Sie Probleme mit den Erklärungen oder Programmen haben, oder wenn ich Ihnen bei anderen offenen Fragen helfen kann: Bitte schreiben Sie mir über den Verlag.

Ich werde Ihnen entweder direkt antworten oder, wenn viele Fragen zusammenkommen, sie im 64'er-Stammheft in einem eigenen Beitrag beantworten.

(Dr. Helmuth Hauck/hm)

Anhang: Befehlsübersicht C16, C64/VC 20, C128

(nach Funktionen geordnet)

Befehl	C16	C64, VC 20	C128
EIN-AUSGABE			
BLOAD			*
BOOT			*
BSAVE			*
CLOSE	*	*	*
CMD	*	*	*
DCLEAR			*
DCLOSE			*
DLOAD	*		*
DOPEN	*		*
DSAVE	*		*
DVERIFY			*
GET	*	*	*
GET #	*	*	*
GETKEY	*		*
INPUT	*	*	*
INPUT #	*	*	*
JOY	*		*
LOAD	*	*	*
OPEN	*	*	*
PEN			*
POT	*		*
PRINT	*	*	*
PRINT #	*	*	*
PRINT USING	*		*
PUDEF	*		*
READ-DATA	*	*	*
RECORD #			*
SAVE	*	*	*
VERIFY	*	*	*
PROGRAMM-ABLAUFSTEUERUNG			
BEGIN			*
BEND			*
DEF-FN	*	*	*
DIM	*	*	*
DO-LOOP-EXIT	*		*
DO-LOOP-UNTIL	*		*
DO-LOOP-WHILE	*		*
END	*	*	*
FOR-TO-NEXT	*	*	*
GOSUB-RETURN	*	*	*
GOTO	*	*	*
GO 64			*
IF-THEN	*	*	*
IF-THEN-ELSE	*		*
LET	*	*	*
MONITOR	*		*

Befehl	C16	C64, VC 20	C128
NEW	*	*	*
ON-GOSUB	*	*	*
ON-GOTO	*	*	*
POINTER			*
REM	*	*	*
RREG			*
RESTORE	*	*	*
RUN	*	*	*
SLEEP			*
SYS	*	*	*
USR	*	*	*
WAIT	*	*	*
SPEICHER			
BANK			*
FETCH			*
FRE	*	*	*
PEEK	*	*	*
POKE	*	*	*
STASH			*
SWAP			*
DISKETTE			
APPEND			*
BACKUP	*		*
CATALOG			*
COLLECT	*		*
CONCAT			*
COPY	*	X	*
DIRECTORY	*		*
HEADER	*		*
INITIALIZE	X	X	X
NEW	X	X	X
RENAME	*	X	*
SCRATCH	*	X	*
VALIDATE	X	X	X

Das Zeichen »X« bedeutet, daß der Disketten-Befehl nur nach der Öffnung eines Befehlskanals mit OPEN 1,8,15 gegeben werden kann.

Befehl	C16	C64, VC 20	C128
GRAFIK			
BOX	*		*
CHAR	*		*
CIRCLE	*		*
COLOR	*		*
DRAW	*		*
GRAPHIC	*		*
LOCATE	*		*
PAINT	*		*
RCLR	*		*
RDOT	*		*
RGR	*		*
RLUM	*		*
SCALE	*		*
WIDTH	*		*
SPRITE-SHAPE			
BUMP			*
COLLISION			*
GSHAPE	*		*
MOVSPR			*
RSPCOLOR			*
RSPPOS			*
RSPRITE			*
SPRCOLOR			*
SPRDEF			*
SPRITE			*
SPRSAV			*
SSHABE	*		*

Befehl	C16	C64, VC 20	C128
TÖNE			
ENVELOPE			*
FILTER			*
PLAY			*
SOUND	*		*
TEMPO			*
VOL	*		*
BILDSCHIRM & FORMATIERUNG			
CLR	*	*	*
POS	*	*	*
RWINDOW			*
SCNCLR	*		*
SPC	*	*	*
TAB	*	*	*
WINDOW			*
FEHLERBEARBEITUNG			
DS	*		*
DS$	*		*
EL	*		*
ER	*		*
ERR$	*		*
RESUME	*		*
ST	*	*	*
TRAP	*		*
STRINGBEARBEITUNG			
ASC	*	*	*
CHR$	*	*	*
INSTR	*		*
LEFT$	*	*	*
LEN	*	*	*
MID$	*	*	*
RIGHT$	*	*	*
STR$	*	*	*
VAL	*	*	*
PROGRAMMIERHILFE			
AUTO	*		*
CONT	*	*	*
DELETE	*		*
FAST			*
HELP	*		*
KEY	*		*
LIST	*	*	*
RENUMBER	*		*
SLOW			*
STOP	*	*	*
TROFF	*		*
TRON	*		*
GEOMETRISCHE & NUMERISCHE FUNKTIONEN			
ABS	*	*	*
ATN	*	*	*
COS	*	*	*
DEC	*		*
EXP	*	*	*
HEX&	*		*
INT	*	*	*
LOG	*	*	*
RND	*	*	*
SIN	*	*	*
SGN	*	*	*
SQR	*	*	*
TI	*	*	*
TI$	*	*	*
BOOLSCHE OPERATIONEN			
AND	*	*	*
NOT	*	*	*
OR	*	*	*
XOR			*

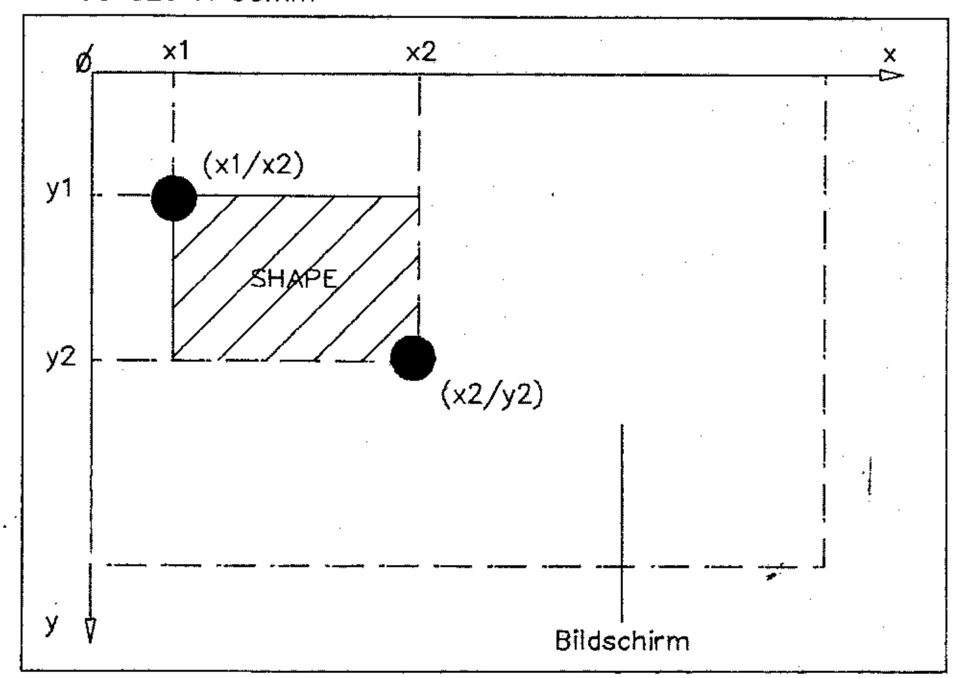


Bild 1. Das schraffierte Rechteck kennzeichnet das spätere Shape

Shapes auf dem C 16

Zwei Grafikbefehle sind es, die manchem C16-Benutzer einige Rätsel aufgeben. Beide haben mit den Shapes zu tun: SSHAPE und GSHAPE werden Ihnen hier vorgestellt.

Eine wichtige Kaufentscheidung bei Commodore-Computern sind ihre ausgezeichneten Grafikfähigkeiten. Doch leider ist ihnen teilweise recht schwer beizukommen. So etwa bei den Shapes. Hier erfahren Sie, wie Sie mit den Grafik-Befehlen SSHAPE und GSHAPE umgehen müssen. Doch zuerst: Was sind Shapes?

Wie so vieles aus der Computerkultur stammt auch dieses Wort aus dem Angelsächsischen: Shape heißt ins Deutsche übersetzt soviel wie Form, Gestalt oder Umriß. Hier bezeichnet Shape einen genau definierten Bildschirmausschnitt, der gespeichert und wiederverwendet werden kann. Wie das mit dem C 16 geschieht, soll nun erklärt werden.

Die Shape-Befehle

1. SSHAPE

Dies ist der Befehl, mit dem ein Bildschirmbereich in einen String eingelesen werden kann. Seine Syntax ist

```
SSHAPE A$,X1,Y1,X2,Y2
```

A$ ist hier eine beliebige Stringvariable, auch ein Array kann Verwendung finden. X1 bis Y2 gibt die Eckkoordinaten des Rechteckes an, dessen Inhalt als Shape definiert werden soll. Bild 1 soll das etwas verdeutlichen.

(X1,Y1) gehören zum linken oberen und (X2,Y2) zum rechten unteren Eckpunkt. Es ist auch möglich, X2 und Y2 wegzulassen. In diesem Fall werden durch den SSHAPE-Befehl einfach die aktuellen Koordinaten des Grafik-Cursors eingesetzt.

Falls es noch unklar sein sollte: Im Gegensatz zu den Sprites anderer Commodore-Computer, die jedem Bildschirm-Modus trotzen (also sowohl als HiRes- oder Multicolorsprites im Text- als auch im Hochauflösungs- und Multicolormodus auftauchen können), sind Shapes fest mit dem aktuellen Modus verbunden. Ein Shape ist ein Grafikobjekt und kann daher auch nur in den Grafikmodi auftreten und definiert werden. Also in denen, die durch GRAPHIC erzeugt werden, wobei n von 1 bis 4 geht. Außerdem ist ein Multicolorshape auch nur im Multicolormodus ein solches. Schaltet man beispielsweise mittels GRAPHIC1 in den normalen Hochauflösungsmodus um, wird auch unser Shape nur als hochaufgelöstes sichtbar.

Wie wir uns aus all dem schon fast denken können, ist das Koordinatensystem, auf das sich die Angaben X1 bis Y2 beziehen, das im jeweiligen Modus gültige. Im Normalfall also im HiRes-Modus X von 0 bis 319 und Y von 0 bis 199. Im Multicolormodus geht X nur von 0 bis 159.

A$ ist ein String. Für Strings gilt beim C 16 die Begrenzung auf maximal 255 Byte. Ein durch SSHAPE in einen String zu schreibendes Gebiet darf also eine gewisse Größe nicht überschreiten. Bezeichnen wir als

DX = X2 - X1 und als

DY = Y - Y1

dann gilt für die Bytemenge im String im Hochauflösungsmodus die Formel:

```
Lh=INT((ABS(DX)+1)/8+.99)*(ABS(DY)+1)+4
```

Im Multicolormodus gilt statt dessen die Gleichung:

```
Lm=INT((ABS(DX)+1)/4+.99)*(ABS(DY)+1)+4
```

Die Addition von 4 am Ende der Formeln ergibt sich durch 4 Byte, die den Schluß des Shape-String, bilden und Steuergrößen enthalten.

Sollte dann, trotz all dieser Berechnungsmöglichkeiten – man wendet ja nicht jedesmal die Formeln an – der zu speichernde Shape größer als 255 Byte werden, dann meldet unser Computer einen STRING TOO LONG ERROR.

Einen Aspekt des SSHAPE-Befehls müssen wir noch besprechen, weil es da einige Verwirrung geben kann: Wie reagiert SSHAPE auf ein durch SCALE verändertes Bild-

Fortsetzung auf Seite 46

Fortsetzung von Seite 44

schirmsystem? Die Koordinatenangabe bei SSHAPE muß ebenfalls in den aktuell gültigen Werten geschehen. Man muß sich also nicht um die Organisation des Koordinatensystems kümmern, denn durch SSHAPE wird automatisch die Umrechnung auf den normalen Bildschirm vorgenommen. Lediglich bei der Berechnung der Stringlänge müssen einige Korrekturen vorgenommen werden.

Nehmen wir an, wir hätten durch SCALE 1 ein System definiert, dann liegen (im Hochauflösungsmodus) die Vergrößerungsfaktoren

FX = 1023/320
FY = 1023/200

vor und die Beziehungen für die Stringlänge lauten dann:

Hochauflösungsmodus

```
Lh=INT((ABS(DX*320/1023)+1)/8+.99)*(ABS(DY*200/1023)
+1)+4
```

Multicolormodus

```
Lm=INT((ABS(DX*160/1023)+1)/4+.99)*(ABS(DY*200/1023)
+1)+4
```

DX und DY sind dabei im aktuellen Koordinatensystem geltende Werte.

Begnügen wir uns nun mit dem SSHAPE-Befehl (es gäbe noch einiges zu untersuchen) und wenden wir uns dem Gegenstück, nämlich dem GSHAPE-Befehl zu.

2. GSHAPE

Ein durch SSHAPE im String gespeichertes Bild kann durch diesen Befehl nun wieder auf dem Bildschirm dargestellt werden:

```
GSHAPE A$,X,Y,M
```

A$ ist der uns nun schon bekannte Speicherstring, der durch SSHAPE definiert wurde. Es kann jede Stringvariable oder ein String-Array-Element verwendet werden.

X,Y sind die Koordinaten im aktuellen System, an die die linke obere Ecke unseres Shapes gelegt werden soll.

Interessant ist die Angabe von M. Hier dreht es sich um einen Darstellungsmodus, von dem es hier 4 Möglichkeiten gibt:

M=0. Unser Shape wird genauso abgebildet, wie es definiert wurde. Schon an dieser Stelle auf dem Bildschirm vorhandene Objekte werden überdeckt. Dasselbe ergibt sich, wenn wir diesen Parameter einfach weglassen.

M=1. Das Shape wird invertiert abgebildet. Auch hier überdeckt das Shape vorhandene Objekte.

M=2. OR-Modus: Bildpunkte werden sichtbar, wenn sie zum Shape oder zu vorhandenen Objekten gehören. Beide Bilder überlagern sich so.

M=3. AND-Modus: Bildpunkte werden nur dann gesetzt, wenn sie sowohl zum Shape als auch zum Bildschirmobjekt gehören.

M=4. EOR- (oder XOR-) Modus: Bildpunkte werden nur dann gesetzt, wenn das entsprechende Bit für Shape und Bildschirmobjekt ungleich ist.

Die Modi 2, 3, und 4 sind etwas schwer vorstellbar. Zur Verdeutlichung finden Sie hier ein kleines Demonstrationsprogramm namens GSHAPE-MODI.

Hier wird zuerst ein Shape definiert, dann der gesamte Bildschirm quergestreift und schließlich das soeben erstellte Shape in allen 5 Modi darauf dargestellt. Die Wirkung ist auf diese Weise ganz gut zu erkennen, besonders bei den Modi 2, 3 und 4.

Der Modus ist aufgrund der EOR-Behandlung interessant: Wird auf dieselbe Stelle noch einmal das Shape im Modus 4 gezeichnet, dann verschwindet es ganz, und der Bildschirmhintergrund ist wieder vorhanden. Will man Shapes bewegen, ohne Bildschirmobjekte zu zerstören, dann ist diese Darstellungsweise ganz gut geeignet.

Die Frage, wie viele Shapes man definieren und auch darstellen könne, erübrigt sich fast: beliebig viele, solange der Speicherplatz für die Strings reicht. Man könnte sich also eine eigene Bibliothek von Hochauflösungs- oder Multicolorobjekten zulegen und diese dann bei Bedarf abrufen. Sollte für ein Objekt der String zu kurz sein, können mehrere Shapes gekoppelt werden. Sie erkennen schon: Auch hier sind die Möglichkeiten sehr breit.

Zusammenspiel von SSHAPE und GSHAPE

Eine Frage haben wir noch offengelassen, die eine recht verlockende Konsequenz in sich birgt: Wenn man mittels der SCALE-Anweisung die Systeme bei SSHAPE (also beim Aufbau des Shape) und bei GSHAPE (bei der Abbildung) unterschiedlich wählt, kann man dann Shapes verkleinern oder vergrößern?

Die Antwort ist leider nein. Bei GSHAPE werden nur die Koordinaten eines (des linken oberen) Eckpunktes angegeben. Somit hat ein unterschiedliches Koordinatensystem lediglich auf den Darstellungsort eine Wirkung, nicht aber auf die Shape-Größe.

Im Gegensatz zu den Sprites der anderen Commodore-Computer ist das Bewegen von Shapes eine langweilige Angelegenheit. Jedem Shape-Aufbau durch GSHAPE kann man ganz geruhsam zusehen. Zwar ist es durch Sequenzen wie

```
FOR = 0 TO 200
GSHAPE A$,I,I,4
GSHAPE A$,I,I,4
NEXT I
```

möglich, Shapes ohne Schaden für den Bildschirminhalt über den Sichtbereich ziehen zu lassen. Das Ganze ähnelt aber bei weitem nicht der Sprite-Bewegung.

Der C16 wird oft unterschätzt. Zwar ist er vom Speicherplatz her – besonders bei eingeschalteter Grafik – wirklich reichlich beschränkt, von seinem Basic her ist er allerdings manchem weitaus kostspieligeren Computer hoch überlegen, was auch diese beiden kleinen Befehle zeigen.

(Heimo Ponnath/dm)

```
10 REM *** DIE GSHAPE-MODI ***
20 REM ERSTELLEN EINES SHAPE
30 COLOR 0,1: COLOR 1,6: COLOR 4,1: GRAPHIC
   1,1
40 CIRCLE 1,10,10,10,10: BOX 1,0,0,20,20: PA
   INT 1,1,1: PAINT 1,19,1: PAINT 1,1,19
50 PAINT 1,19,19: CIRCLE 1,10,10,7,7: DRAW 1
   ,10,0 TO 10,20: DRAW 1,0,10 TO 20,10
60 SSHAPE A$,0,0,20,20
70 SCNCLR
80 REM BILDSCHIRMHINTERGRUND
90 DX=5: DY=5: X=0: Y=0
100 DO WHILE X<511
110 : X=X+DX: Y=Y+DY
120 : DRAW 1,0,Y TO X,0
130 LOOP
140 REM ABBILDEN DES SHAPE
150 FOR I=0 TO 4
160 : GSHAPE A$,10+55*I,100,I
170 NEXT I
180 REM KOMMENTAR
190 COLOR 1,3
200 FOR I=0 TO 4
210 : B$=RIGHT$(STR$(I),1)
220 : CHAR 1,2+7*I,18,B$
230 NEXT I
240 CHAR 1,1,4,"DIE GSHAPE-MODI UND IHRE WIR
    KUNGEN:",1
250 CHAR 1,1,23,"BITTE EINE TASTE"
260 GET KEY A$
270 GRAPHIC 0,1
280 END
```

Listing 1. Die verschiedenen Modi beim GSHAPE-Befehl und ihre Wirkungen

Das Auge »ißt« mit

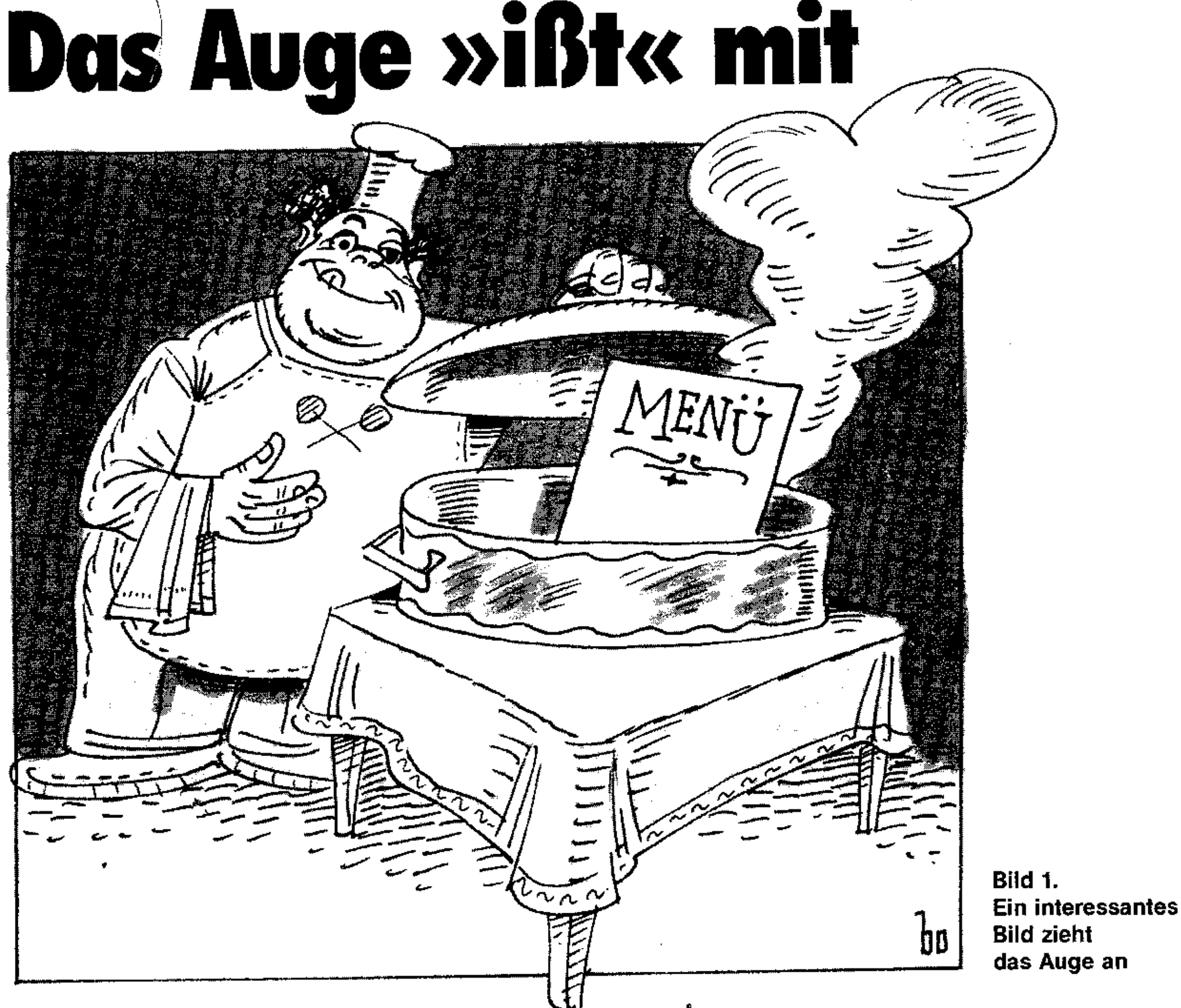


Bild 1. Ein interessantes Bild zieht das Auge an

Genauso wie beim Essen das Auge mitißt, steigert auch das optische Bild (Menü) eines Programmes das Interesse. Wie Sie diesen Appetitanreger in Ihr Programm einbinden können, erfahren Sie in diesem Kurs.

Viele Programme zeichnen sich dadurch aus, daß sie eine sehr anwenderfreundliche Form der Menüsteuerung (Bild 1) besitzen. Sieht man sich einige Menüs von Programmen kommerzieller Software-Hersteller an, wie zum Beispiel Textverarbeitungs- oder Dateiverwaltungsprogramme, so kann man hier mit den Cursortasten die einzelnen Programmfunktionen auswählen. Dabei werden mit den Cursortasten einzelne Menüpunkte hervorgehoben. Ist der gewünschte Menüpunkt erreicht, wird er mit einer Taste, zum Beispiel die F1-Taste, aktiviert.

Die einfachste Form der Menüsteuerung ist die, die einzelnen Menüpunkte durchzunumerieren. Mit der Basic-Anweisung »ON n GOTO« können die Unterprogramme sehr schnell aufgerufen werden. Zum Beispiel:

```
100 INPUT" MENUEPUNKT";A
110 ON A GOTO 1000,2000,3000,...,n
```

Eine solche Menüsteuerung wird in Listing 1 angewandt. Dieses Programm ermöglicht es, die Werte »SPIELSTUFE«, »GESCHWINDIGKEIT« und »ANZAHL DER SPIELER« zu verändern. Die Werte »SPIELSTUFE« und »GESCHWINDIGKEIT« können 1, 2 oder 3 annehmen, »ANZAHL DER SPIELER« 1 oder 2. Ein solches Menü könnte vor einem Spielprogramm stehen.

Über einen INPUT-Befehl (Zeile 320) wird die Nummer des gewünschten Menüpunktes eingegeben, auf Richtigkeit der Eingabe wird in Zeile 330 geprüft (es sind nur 1, 2, 3 oder 4 als Eingaben gestattet) und in der Zeile 360 in das Unterprogramm gesprungen. Der Befehl

```
360 ON P GOTO 380, 450, 520, 590
```

verzweigt, wenn die Variable P, die in Zeile 320 die Nummer des gewünschten Menüpunktes vom Benutzer über die INPUT-Anweisung (320 INPUT "..";P) erhält, den Wert 1 hat, nach 380, bei P=2 nach 450 und so weiter.

Die Adressen, die hinter dem ON n GOTO-Befehl stehen, zeigen jeweils auf die erste Zeilennummer einer Unterroutine, die vorher per Menü ausgewählt wurde. Zu den Zeilen 370 bis 620 ist nichts weiter zu sagen, als daß dort die Unterroutinen stehen, die die Eingabe von Werten beziehungsweise das Verlassen des Menüs bewirken.

Die Variablen L, G und A enthalten die Werte für »Spielstufe« (<L> für »Level«), »Geschwindigkeit« <G> und »Anzahl der Spieler« <A>. Die Variable P enthält die Nummer des Menüpunktes, der ausgeführt werden soll.

Diese Form des Menüs, die in Listing 1 verwendet wird, ist allerdings noch recht unkomfortabel. Störend ist, daß jede Eingabe der Menünummer immer mit der RETURN-Taste bestätigt werden muß. Im folgenden soll dies geändert werden. Dies erreicht man durch Verwendung einer GET- beziehungsweise GETKEY-Abfrage anstelle des INPUT-Befehls, wie in Listing 2 in Zeile 640 zu sehen ist. Listing 2 hat jedoch noch eine weitere kleine Ergänzung, die ich Ihnen hier nebenbei vorstellen möchte, weil sie simpel, somit leicht verständlich und ohne nennenswerte Probleme programmierbar und wirkungsvoll zugleich ist: die Window-Technik.

Damit Sie sofort erkennen können, worum es sich handelt, sollten Sie Listing 2 abtippen, starten und alle Funktionen ausprobieren. Jetzt drücken Sie nur noch 1, 2, 3 oder 4 (je nach Menüpunkt), dann können Sie gegebenenfalls einen

der Werte »Spielstufe«, »Tempo« (=»Geschwindigkeit« aus Listing 1) und »Anzahl der Spieler« ändern. Diese verbesserte Form der Parametereingabe hat allerdings nichts mit der Window-Technik zu tun, mit der wir uns nun beschäftigen wollen.

Ein Window (»window« = englisches Wort für »Fenster«) ist ein Bildschirmausschnitt, der als Teilbildschirm behandelt wird und für einen bestimmten Zweck (zum Beispiel die Anzeige eines Copyrights) reserviert ist. Mit Hilfe der Sequenzen ESC-B beziehungsweise ESC-T können Sie beim C 16/116 das Window bestimmen, es handelt sich jedoch nur um eine Verkleinerung des Bildschirmbereiches: dieser Teilausschnitt wird wie der gesamte Bildschirm gehandhabt. Dies ist nützlich, um Teile des Bildschirms vor dem Überschreiben zu schützen (für eigene Programme äußerst unbrauchbar). Im Beispielprogramm (Listing 2) sind einzelne Teile des Bildschirms für bestimmte Aufgaben freigehalten; so gibt es:
- ein Copyright-Window, in dem der Copyright-Vermerk steht,
- ein Level-Window, in dem die Spielstufe steht,
- ein Tempo-Window, in dem die Geschwindigkeit steht,
- und ein Anzahl-Window, in dem die Anzahl der Spieler steht.

Die Windows werden in den Zeilen 290 bis 590 ausgegeben, und zwar dann, wenn auf eine neue Eingabe gewartet wird. Da die Windows immer an derselben Bildschirmposition stehen und nicht verschoben werden, muß vor der Neuausgabe eines Windows nicht erst der Bildschirm gelöscht werden. Dieser Trick, immer nur deckungsgleiche Bereiche auszugeben (wodurch dem Benutzer der Neuaufbau eines Bildschirmbereichs nicht auffällt), wird an späterer Stelle in diesem Kurs – bei einer anderen Form der Menüsteuerung – erklärt. Dort können Sie dann die Funktionsweise erfahren, die uns hier im Moment nicht weiter aufhalten soll.

Bei der Verwendung von Windows sind folgende Punkte zu beachten:

1. An der Stelle, an der ein Window stehen soll, darf vorher kein anderer Text stehen, da dieser sonst durch das Window überschrieben wird. Wenn Sie einmal sehen wollen, wie ein Window einen Teil des Bildschirms überschreibt, ändern Sie jeweils die drei Parameter nach CHAR in den Zeilen 340 bis 390 von 2 bis 7 auf 7 bis 12. Starten Sie dann das Programm, und das Copyright-Window überschreibt die Menüpunkte teilweise.

2. Windows muß man optisch abgrenzen, damit sie vom »normalen« Bildschirm unterschieden werden können. In der Regel nimmt man Grafikzeichen, die das Window einrahmen. In Listing 2 sehen Sie einige verschiedene Möglichkeiten der Eingrenzung eines Windows (durch Reversdruck, Grafikzeichen, Sterne), und es gibt noch mehr Variationen.

3. Die Position des Windows muß festgelegt sein. Vor dem Ausdrucken eines Windows muß diese Window-Position als Cursorposition eingestellt werden. Hierbei ist der CHAR-Befehl eine große Hilfe. Dieser Befehl kann nämlich nicht nur Texte in hochauflösende Grafik schreiben, sondern auch wie ein PRINT-Befehl bestimmte Stellen im normalen Text-Modus beschreiben. Die Syntax für CHAR zum PRINTen an eine beliebige Cursorposition ist:

```
CHAR 1, Spalte,Zeile,"TEXT ALS STRING"
```

Die »1« hinter CHAR gibt an, daß in die normale Textseite geschrieben werden soll. Spalte und Zeile entspricht der Cursorposition, bei der die Textausgabe beginnen soll. »TEXT ALS STRING« ist der auszugebende Text.

Ein Tip: In der Regel wird man den Wert für »Spalte« beim Drucken eines Windows nicht ändern, da der linke Rand nicht verändert wird; der Wert für »Zeile« ist jedoch dauernd um 1 hochzuzählen, wie man unter anderem an den Zeilen 340 bis 390 verfolgen kann (3. Parameter nach CHAR!).

4. Der Einfachheit halber sollte man nur Windows in Rechteckform programmieren, weil man in diesem Fall nur zwei Begrenzungen des Windows kennen muß: Länge und Breite. Da das C 16/116-Betriebssystem das Arbeiten mit Windows, wie wir es benötigen würden, nicht unterstützt, muß man auf die Einhaltung dieser Grenzen selbst achten. Bei Rechteck-Windows der gebräuchlichen Form muß man folgende Werte festlegen:
- linke Grenze
- rechte Grenze
- obere Grenze
- untere Grenze

Man darf also, weil ein Window kleiner als der gesamte Bildschirm ist, nur eine begrenzte Textbreite verwenden; außerdem stehen nicht beliebig viele Zeilen zur Verfügung. Dies ist ganz einfach eine Frage der Planung beim Schreiben der PRINT- beziehungsweise CHAR-Zeilen. Betrachten wir dazu einmal die Zeilen 340 bis 390. Jeder dieser sechs CHAR-Befehle hat als 2. Parameter eine 5, das Drucken wird also immer in der linken Spalte begonnen. Damit ist die Einhaltung der linken Grenze garantiert. Die obere Grenze wird dadurch eingehalten, daß die Ausgabe in der 2. Zeile beginnt (Zeile 340: CHAR .,.,2,...), die untere Grenze dadurch, daß nicht weiter als bis zur 7. Zeile gedruckt wird (Zeile 390: CHAR .,.,7,...).

Die rechte Grenze einzuhalten ist am schwierigsten, denn hier kommt es nicht nur auf die Parameter des CHAR-Befehls an. Wenn man aber eine einheitliche Länge des Ausgabe-Strings einhält – in den Zeilen 340 bis 390 hat jeder Ausgabe-String exakt 11 Zeichen Länge, das Window ist also 11 Zeichen breit – dürfte es keine Probleme geben. Wir können nur abschließend wiederholen: Wenn man beim Programmieren ausreichend genau vorgeht, sind Windows ein Kinderspiel.

Wir hoffen, daß Ihnen der Ausflug ins Thema »Windows« Spaß gemacht hat; nun wollen wir uns bis zum Ende dieses Kurses mit einer ähnlich ästhetischen Form der Menüsteuerung befassen. Dazu ist zunächst etwas vorauszuschicken.

Der C 16/116 bietet im Gegensatz zum VC 20/C 64 eine Funktionstastenbelegung, die Sie als C 16/116er sicher zu schätzen wissen. Der einzige Haken dieser Funktionstastenbelegung ist jedoch, daß die Abfrage der Funktionstasten über GET/GETKEY erschwert wird. Um dies zu umgehen, ist es erforderlich, den C 64/VC 20-Zustand wiederherzustellen, indem die Funktionstasten wie Steuertasten (CLEAR, HOME, Cursortasten, Farbtasten,...) gehandhabt werden. Zwar kann man mit den auf die Funktionstasten gelegten Steuertasten keine Steuerfunktion hervorrufen (die verwendeten Codes 133 bis 140 haben eben keine Funktion), aber man kann leicht auf eine Funktionstaste warten:

```
10 KEY 1,CHR$(133):REM F1-Taste mit Code belegen
20 GETKEY A$:IF A$< >CHR$(133) THEN 20:REM auf F1-
Code warten
```

Von den Steuertasten wie CLEAR, HOME, Cursortasten und so weiter kennen Sie die Möglichkeit, Steuerzeichen in Anführungszeichen als reverse Symbole zu schreiben (siehe Eingabehinweise für Basic-Programme in diesem Heft). Nach der Belegung der Funktionstasten mit Steuercodes geht dies auch mit F-Tasten. Nach Eingabe von KEY 1,CHR$(133) ist die F1-Taste mit dem Code 133 belegt; durch einmaliges Drücken der Anführungszeichen-Taste wird die F1-Taste als reverses Grafikzeichen abgebildet, wie es innerhalb von Anführungszeichen auch mit der HOME-Taste geschieht.

Alle nun folgenden Listings machen von der Abfrage der F1- und der F7-Taste (die anderen Funktionstasten bleiben unberührt) Gebrauch. Dadurch steht zwar auf F1 nicht mehr »GRAPHIC« und auf F7 nicht mehr »LIST«, aber wir können die Tasten abfragen. Die erforderlichen KEY-Befehle stehen in jedem der nun folgenden Listings, müssen jedoch schon

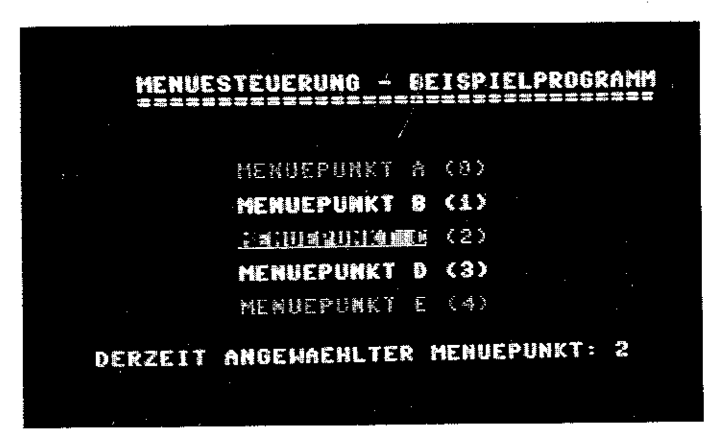


Bild 2. Das Menü aus Listing 3

vor der Eingabe der Programme eingegeben werden, um in Anführungszeichen die F1/F7-Taste mit einem Steuerzeichen auszudrücken. Geben Sie also vor der Eingabe aller folgenden Listings dieses Kurses die Anweisung

```
KEY 1,CHR$(133):KEY 7,CHR$(136)
```

im Direktmodus ein. Wenn Sie ein solches gespeichertes Programm laden wollen, sind diese Befehle nicht erforderlich, da sie bereits im Programm stehen.

Damit Sie gleich in den Genuß eines guten Menüs kommen, tippen Sie einfach Listing 3 ab und starten es. Der Bildschirm wird dunkel und ein Menü mit den Punkten A bis E erscheint. Hinter jedem Menüpunkt steht eine Zahl in Klammern, die die Nummer des Unterprogramms angibt, das Sie anwählen können. Die Funktionen des kleinen Programms im einzelnen:

Mit <CURSOR DOWN> kommen Sie zum nächsten Punkt.

Mit <CURSOR UP> erreichen Sie den vorhergehenden Punkt.

Vom untersten Menüpunkt aus können Sie mit CURSOR DOWN direkt in den obersten und mit CURSOR UP vom obersten Punkt zum untersten springen.

Mit HOME können Sie von jedem Punkt immer in den obersten springen.

<F1> aktiviert das angewählte Unterprogramm.

<F7> bricht das Programm mit END ab.

Probieren Sie ohne weiteres alle Funktionen von Listing 3 aus (Bild 2). Sollte Ihnen diese Art der Menüsteuerung gefallen haben, dann erfahren Sie im folgenden mehr über den Aufbau.

Im Bild 3 finden Sie die schematische Darstellung der Menüpunkte von Listing 3. Die Randpunkte A (0) und E (4) sind besonders hervorgehoben, weil an diesen die Rotation stattfindet.

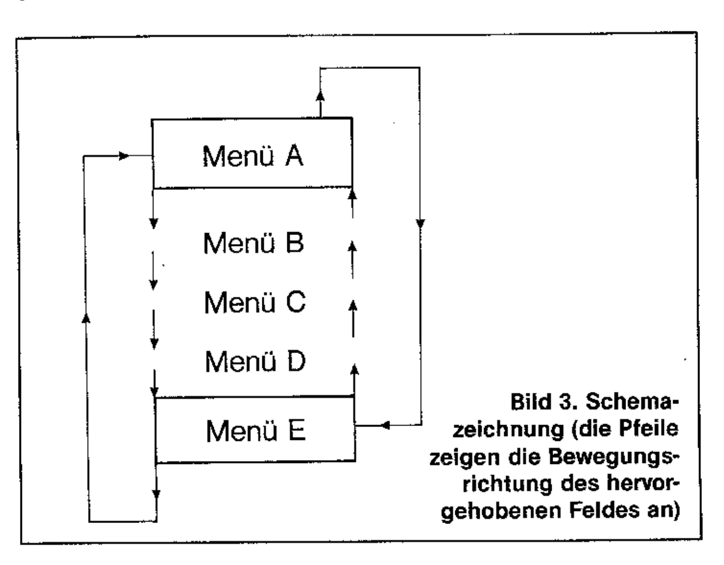


Bild 3. Schemazeichnung (die Pfeile zeigen die Bewegungsrichtung des hervorgehobenen Feldes an)

Die Variablen von Listing 3:

A (Menüanzahl) enthält die Anzahl der Menüpunkte (5). A ändert sich nie und ist folglich eine Konstante.

R$(n) ist immer ein 1-Byte-String, der den Wert CHR$(0) enthält. Der Code CHR$(0) steht für Null und hat keine Funktion. Wenn ein Menüpunkt hervorgehoben is, so steht in R$(n) der Wert CHR$(18) für <REVERS ON>.

P (Programmpunkt) enthält die Nummer des angewählten Unterprogramms. P wird durch die Cursortasten geändert und gibt den Menüpunkt an, der hervorgehoben werden soll. Des weiteren wird P zur Anwahl der Unterprogramme benötigt. (490 ON P+1 GOTO 610, 680, ...)

T$ (Taste) In der Zeile 480 wird die Tastatur abgefragt. Der Tastencode wird in der Variablen T$ gespeichert. In den Zeilen 490 bis 500 und 520 bis 540 wird der Tastencode mit Bedingungen verglichen. Ist die Bedingung wahr, so wird die nachfolgende Anweisung ausgeführt.

Nachdem die Variablen beschrieben wurden, soll nun der Programmaufbau betrachtet werden.

Zeile 200 bis 300

In der Zeile 250 werden die Variablen definiert. Nach der DIM-Anweisung in der Zeile 250 ist übrigens das gesamte Feld R$(n) mit Leerstrings belegt. (Sollten Ihnen einige Begriffe, zum Beispiel »Feld« nicht bekannt sein, so schauen Sie sich doch bitte den Basic-Kurs oder das Lexikon an.)

In der Zeile 260 wird die Variable P (Programmpunkt) auf Null gesetzt (Menüpunkt A).

Die Anweisung R$(P) = CHR$(18) in Zeile 270 hebt den Menüpunkt hervor, indem der Zeichenhintergrund die Farbe des Zeichens erhält und das Zeichen in Schwarz dargestellt wird. Im folgenden werde ich diese Darstellung als invers bezeichnen.

Schließlich werden in Zeile 280 die Bildschirm- und Rahmenfarbe auf Schwarz gesetzt, der Groß-/Grafikmodus aktiviert und der Bildschirm mit PRINT CHR$(147) gelöscht.

Hauptschleife: Zeile 310 bis 560

Bei der Zeile 310 beginnt die Hauptschleife. Das Menü springt immer von Zeile 560 aus auf 310 zurück, solange kein Unterprogramm aktiviert wurde.

Textausdruck: Zeile 340 bis 420

In den Zeilen 340 bis 420 findet man die Textausgabe der fünf Menüpunkte. (Die Zeile 340 wird anschließend noch gesondert besprochen). Vor den einzelnen Menüpunkten steht die Stringvariable R$(n):

```
380 PRINT TAB(9);R$(0);"{Farbe}Menuepunkt A {Revers off}(0){CURSOR DOWN}"
```

R$(n) wirkt hier als Schalter. Ist R$(n)=CHR$(0), so schaltet er den Reversmodus nicht ein. Ist dagegen R$(n)=CHR$(18), so schaltet er den Modus ein. Die Zeile 430 gibt das gerade angewählte Programm (Nummer) an.

Warten auf Tastendruck: Zeile 450 bis 480

Diese Programmzeile holt ein Zeichen von der Tastatur und legt sie in der String-Variablen T$ ab. Trifft die Bedingung T$ = Leerstring zu, so verzweigt das Programm zurück zum Anfang dieser Zeile. Damit wartet das Programm in dieser Schleife, bis eine Taste gedrückt wurde.

```
480 GET T$ : IF T$ = "" THEN 480
```

Programmaufruf: Zeile 490

In der Zeile 490 wird geprüft, ob die zuletzt gedrückte Taste die Funktionstaste F1 war. Ist diese Verknüpfung wahr, so wird die Anweisung, die hinter THEN steht, ausgeführt.

```
490 IF T$="{F1}" THEN ON P + 1 GOTO 610, 680, 730
```

Dabei bedeutet die Anweisung ON P GOTO (Zeile): springe auf die Zeilennummer, die durch den Zeiger P angewählt ist. Ein Beispiel: P enthält den Wert 2. Zu diesem wird laut Anwei-

sung eine 1 addiert (P + 1) und ist damit 3. Dieser Wert gibt die Position (3) in der Tabelle an, an der die Zeilennummer steht, zu der gesprungen werden soll. In unserem Beispiel wäre es die Zeile 730.

Programmabbruch: Zeile 500

Ergibt die Prüfung `T$ = "{F7}"`, daß diese Bedingung zutrifft, so wird das Programm mit einer Bemerkung beendet.

```
500 IF T$ = "{F7}" THEN PRINT "{CURSOR 2 mal DOWN}
Programmabbruch." : END
```

Rücksetzen: Zeile 510

Es wird die Stringvariable R$(P), die an dieser Stelle noch den Wert CHR$(18) enthält, auf CHR$(0) geändert.

```
510 R$(P) = CHR$(0)
```

Programmanwahl: Zeile 520 bis 540

In diesen drei Zeilen wird geprüft, ob die Taste CURSOR UP, CURSOR DOWN oder HOME gedrückt wurde. (Auf diese Anweisungen wird noch gesondert eingegangen.)

Menüpunkt invertieren: Zeile 550

In der Zeile 550 wird der Variablen R$(P), wobei P jetzt den neuen Menüpunkt beinhaltet, der Wert CHR$(18) zugewiesen.

```
550 R$(P) = CHR$(18)
```

Rücksprung: Zeile 560

Hier ist die Hauptschleife beendet und es wird auf die Zeile 310 zurückgesprungen.

```
560 GOTO 310
```

Unterprogramme: ab 570

Es folgen die fünf Unterprogramme, die Sie vom Menü aus aufrufen können.

Die Struktur des ersten Programms wäre damit, bis auf einige Besonderheiten, beschrieben. Um die Bewegung des Reversfeldes zu simulieren, muß das alte Bild bei jedem Tastendruck mit dem neuen Bild überschrieben werden. Die Frage ist: wie erreicht man es, daß sich die Bilder absolut genau decken? Man legt dazu einen Bezugspunkt, die linke obere Ecke, auf dem Bildschirm fest. Der Basic-Befehl dazu ist HOME oder PRINT CHR$(19). Fügt man den Befehl als PRINT CHR$(19)-Anweisung in ein Programm ein, so springt der Cursor in die linke obere Ecke des Bildschirms. Läßt man nun den Bildschirminhalt erneut ausdrucken, so liegt er deckungsgleich zum alten Bild. Die Anweisung PRINT CHR$(19) kann auch durch die Sequenz, PRINT "{HOME}", ausgedrückt werden.

Ersetzen Sie in der Zeile 340 die Anweisung CHR$(19) durch CHR$(147) (Bildschirm löschen), so wird der Neuaufbau des Bildes deutlicher, weil der Bildschirm jedesmal komplett gelöscht wird. Dies macht sich dann durch ein Flackern des Bildes bemerkbar.

Die eigentliche Besonderheit des kleinen Programms steht in den Zeilen 520 bis 540. Mit der Zeile 540, der einfachsten, soll begonnen werden.

```
540 IF T$ = "{HOME}" THEN P = 0
```

Ist die Bedingung `T$ = "{HOME}"` wahr, so wird die Variable auf 0 gesetzt. Damit enthält P die Nummer des obersten Menüpunktes und druckt diesen in inverser Schrift aus.

Betrachtet man noch einmal Bild 3, so erkennt man, daß das Inversfeld jeweils um 1 nach oben wandert, wenn die CURSOR UP-Taste gedrückt wird. Zum Beispiel wandert das Feld von C nach B und der Wert für P sinkt um 1, von 3 auf 2. Daher heißt es auch in der Zeile:

```
530 IF T$ = "{CURSOR DOWN}" THEN P = P - 1 ...
```

Hatte die Variable vorher den Wert 0, so vermindert sie sich jetzt auf den Wert (-1). Ein Menüpunkt mit der Nummer (-1) existiert aber nicht, statt dessen soll vielmehr von dem Menüpunkt A aus direkt nach Menü E gesprungen werden. Es folgen nun mehrere Möglichkeiten, mit denen diese Bedingung erfüllt werden kann. Ein entsprechendes Flußdiagramm finden Sie im Bild 4.

Der erste Weg:

P = P - 1 : IF P = (-1) THEN P =

oder

P = P - 1 : IF P = (-1) THEN P = P + 5

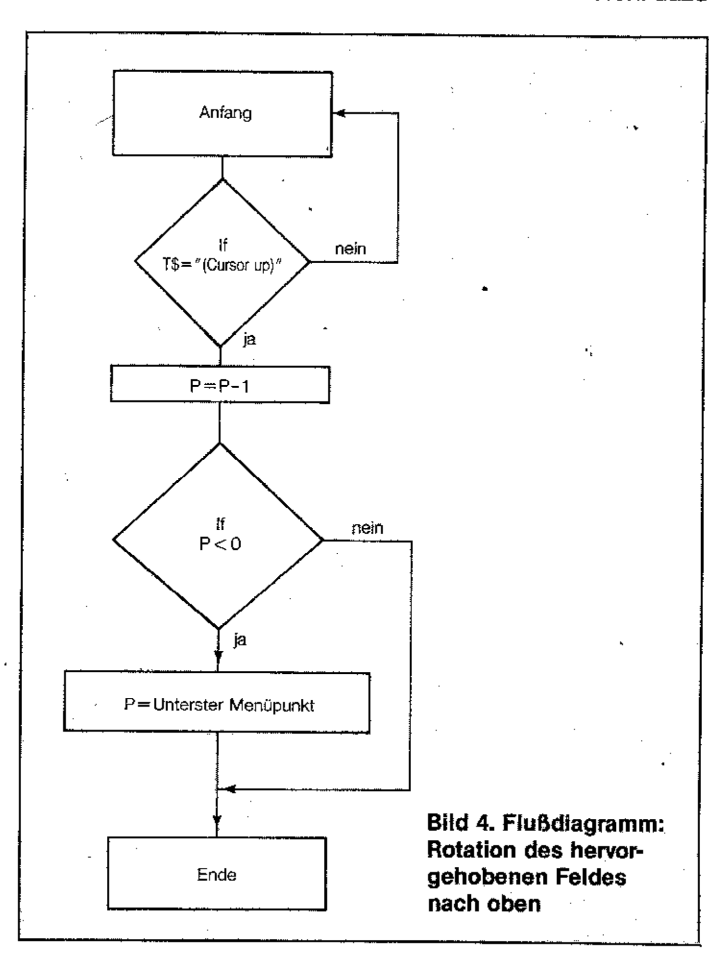


Bild 4. Flußdiagramm: Rotation des hervorgehobenen Feldes nach oben

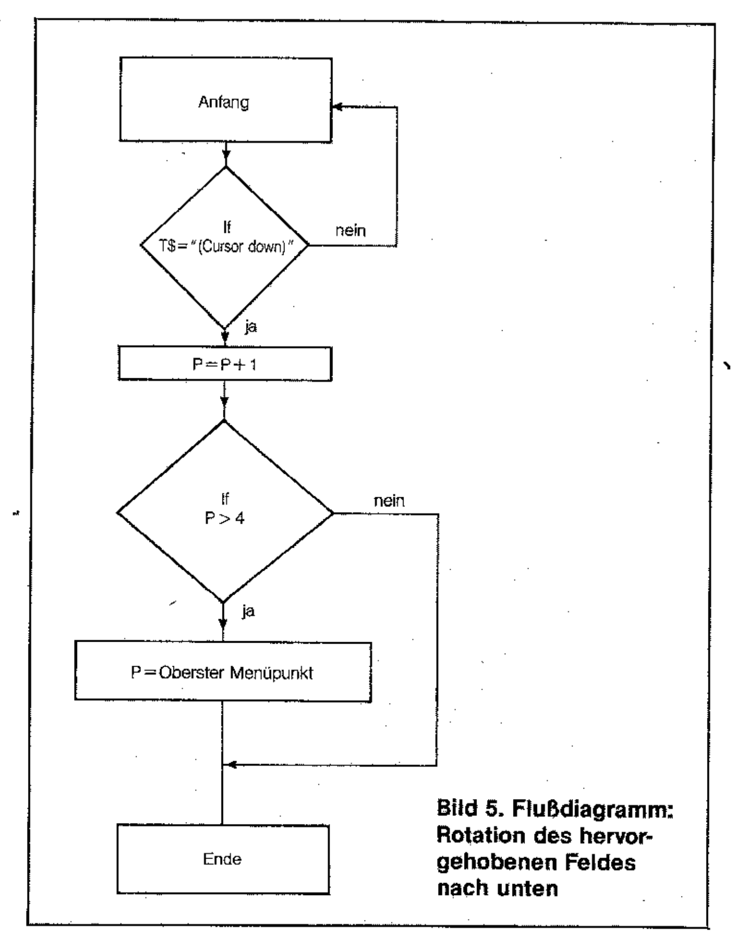


Bild 5. Flußdiagramm: Rotation des hervorgehobenen Feldes nach unten

MENUESTEUERUNG - BEISPIELPROGRAMM

DIESMAL SIND DIE MENUEPUNKTE HORIZONTAL ANGEORDNET.

PUNKT A PUNKT B PUNKT C PUNKT D PUNKT E

Bild 6. Das horizontale Menü aus Listing 5

MENUESTEUERUNG - BEISPIELPROGRAMM

MENUEPUNKTE SIND HORIZONTAL & VERTIKAL ANGEORDNET.

PUNKT A PUNKT D
PUNKT B PUNKT E
PUNKT C PUNKT F

Bild 7. Das zweidimensionale Menü aus Listing 6

Der Weg, der beschritten wurde, ist eigentlich deutlich. Es wird das Ergebnis aus der Subtraktion P – 1 mit dem Wert (–1) verglichen. Ergibt der Vergleich, daß P = (–1) ist, dann wird der Wert der Variablen P durch 4 ersetzt oder um 5 vermehrt.

In der Zeile 530 steht dagegen:

P = P – 1 – A * (P = 0)

Hier wird zunächst auch der Wert der Variablen P um 1 vermindert. Ist die in Klammern stehende Bedingung (P = 0) wahr, das heißt gleich, so wird der Ausdruck durch (–1) ersetzt. Dieser Wert kann nun in die Rechnung einbezogen werden. Die Gleichung lautet dann:

```
  ...THEN P = P - 1 - A * (P = 0)
=→...THEN P = P - 1 - A * (-1)
=→...THEN P = P - 1 - (-A)
=→...THEN P = P - 1 + A
```

Ersetzen Sie die Konstante A durch 5 und die Variable P durch 0, so erhalten Sie folgende Gleichung:

```
  ...THEN P = 0 - 1 - 5 * (0 = 0)
=→...THEN P = 0 - 1 - 5 * (-1)
=→...THEN P = 0 - 1 - (-5)
=→...THEN P = 0 - 1 + 5
=→...THEN P = 4
```

Ist die Bedingung P = 0 nicht wahr, so wird vereinbarungsgemäß eine 0 für die Antwort »das ist falsch« geschrieben. Ersetzt man in der Rechnung den Ausdruck (P = 0) durch unwahr (0), so ändert sich die Gleichung in

```
  ...THEN P = P - 1 - A * (P = 0)
=→...THEN P = P - 1 - A * (0)
=→...THEN P = P - 1 - 0
=→...THEN P = P - 1
```

Kontrollieren Sie diese Gleichung wieder mit folgenden Werten: A = 5 und P = 2.

Betrachtet man nun die Zeile 520, so ähnelt sie der Zeile 530. Deshalb kann die Abhandlung auch etwas kürzer ausfallen (Bild 5).

In der Zeile 520 steht:

...P = P + 1 + A * (P = (A – 1))

Auch hier ist wieder die Verknüpfung (P = (A–1)) der Hauptbestandteil. Geht man davon aus, daß P = 4 ist, so muß die Rechnung wie folgt aussehen.

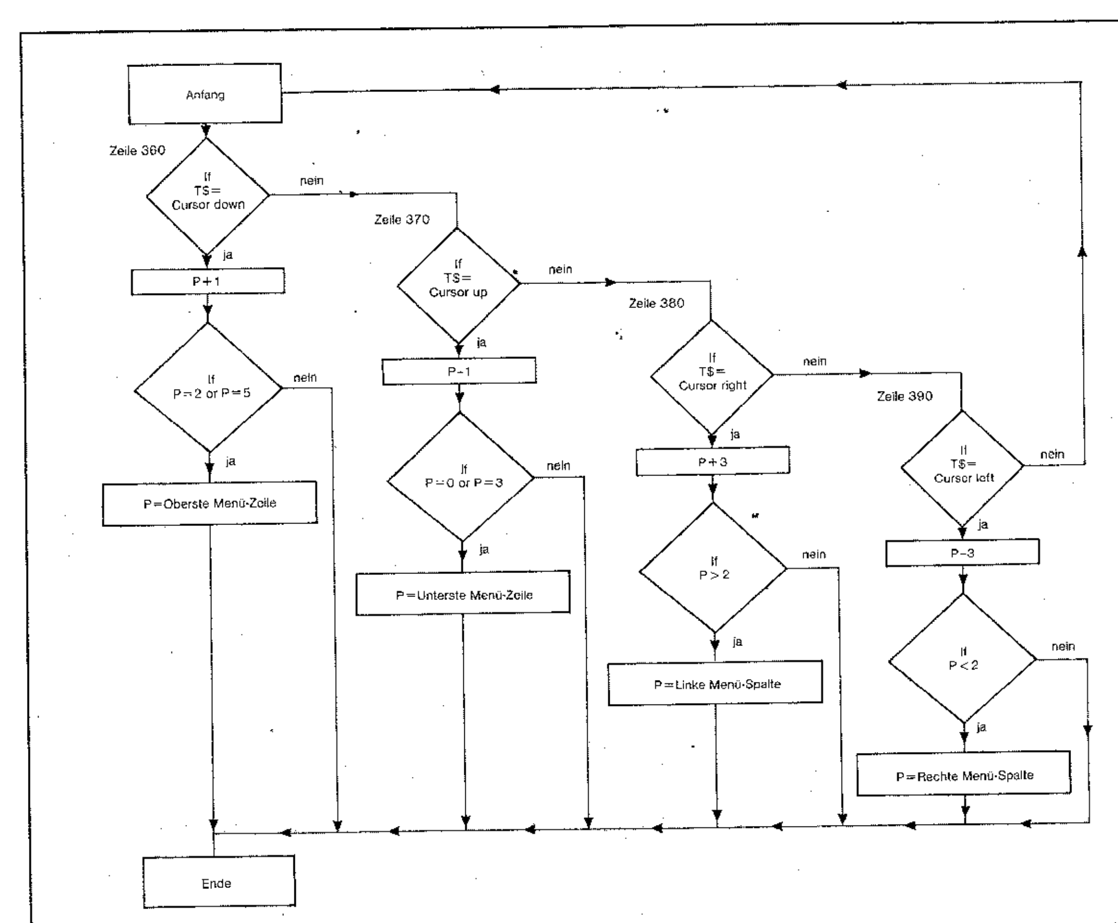


Bild 8. Flußdiagramm: Verschiebung des Inversfeldes in zwei Ebenen

```
...THEN P = P + 1 + A * (P = (A - 1))
=→...THEN P = P + 1 + A * (-1)
=→...THEN P = P + 1 + (-A)
```

Da P = 5-1 wahr ist, sieht die mit Zahlen ersetzte Gleichung so aus:

```
...THEN P = 4 + 1 + 5 * (4 = (5 - 1))
=→...THEN P = 4 + 1 + 5 * (-1)
=→...THEN P = 4 + 1 + (-5)
=→...THEN P = 0
```

Ich nehme an, daß Ihnen der Algorithmus klar geworden ist. Im Anhang finden Sie noch weitere Erklärungen zur Computerlogik.

Warum nun diese umständliche Schreibweise? Alle IF-Anweisungen benötigen viel Rechenzeit, mathematische Operationen dagegen sind verhältnismäßig schnell. Deshalb lohnt sich die Verwendung dieses Algorithmus in allen Vergleichsverfahren, die durch eine Rechnung ersetzt werden können.

In der untersten Zeile unseres Menüs steht die Anmerkung »Derzeit angewählter Menüpunkt«. Dahinter wurde bisher die Nummer des Menüpunktes angezeigt. Dagegen sind alle Menüpunkte im Text durch einen Buchstaben gekennzeichnet (Bild 2). Deshalb wird die Zeile 430 geändert in:

```
430 PRINT "DERZEIT ANGEWAEHLTER MENUEPUNKT:" CHR$
(65 + P)
```

Der Code für den Buchstaben »A« ist die 65 (siehe auch im Commodore-Handbuch ASCII-Code). Alle im Alphabet folgenden Buchstaben haben jeweils eine um 1 größere Codezahl. Auf Grund dieser Tatsache läßt sich der Buchstabe des Menüpunktes mit der kleinen Rechnung

Code = 65 + P

ermitteln.

Listing 2 ist eine Variante von Listing 3, wie sie in der Praxis angewendet wird. Die Informationen bezüglich der internen Numerierung sowie die REM-Zeilen wurden weggelassen. Außerdem wurden die Zeilen teilweise komprimiert. So steht für PRINT CHR$(0) jetzt PRINT " " und für PRINT CHR$(18) jetzt PRINT "{Steuertaste}". Die wichtigste Änderung dagegen ist, daß die Konstante A entfällt und dafür alle Ausdrücke, in denen A vorkommt, ausgerechnet werden. (Vergleichen Sie die Zeilen 520 und 530 von Listing 3 mit 360 und 370 von Listing 4.)

Warum nicht horizontal?

Bislang wurden die Menüs untereinander angeordnet, man könnte auch von einem vertikalen Menü sprechen. Es ist aber kein Problem, die Menüs auch horizontal (waagerecht) anzulegen. In Bild 6 sehen Sie, wie dies gemeint ist. An den Punkten A und E erfolgt wieder die Rotation. Bei einem horizontalen Menü entspricht die Bewegung des Inversfeldes nach rechts die der Bewegung nach unten im vertikalen Menü. Die Bewegung nach links entspricht der nach oben. In Listing 5 finden Sie ein horizontales Menü, wie Sie es in Bild 6 sehen. Die mit der Taste F1 aufgerufenen Unterprogramme geben den Inhalt der Variablen P an. Wie Sie in Zeile 360 und 370 sehen, wird die Funktion der CURSOR DOWN-Taste von der CURSOR RIGHT-Taste übernommen. Ebenso ersetzt die CURSOR LEFT-Taste jetzt die CURSOR UP-Taste. Der größte Teil von Listing 5 wurde aus dem Listing 4 übernommen. Außer der Tastaturabfrage haben sich nur die Printanweisungen geändert. Zusätzlich sind die Zeilen 410 bis 580 (Unterprogramme) durch die Anweisung

```
330 IF T$ = "{F1}" THEN PRINT "ANGEWAEHLTER PUNKT:"; P
: END
```

ersetzt worden.

Zum Abschluß des Kurses soll noch das vertikale und horizontale Menü miteinander verbunden werden. Es entsteht ein zweidimensionales Menü (Listing 6). In den mehrdimensionalen Menüs werden alle vier Cursortasten mit einbezogen. Die CURSOR UP- und -DOWN-Tasten werden im Prinzip wie im vertikalen Menü (siehe Listing 3) zur Ansteuerung der vertikalen Ebenen (A-D, B-E, C-F siehe Bild 7) verwendet. Die CURSOR LEFT- und -RIGHT-Tasten werden zur Anwahl der beiden Spalten verwendet. Die Flußdiagramme finden Sie in Bild 8. Die entsprechenden Berechnungen stehen in den Zeilen 360 bis 390.

```
360 IF T$ = "{CURSOR DOWN}" THEN P = P + 1 + 3*
(P = 2 OR P = 5)
370 IF T$ = "{CURSOR UP}" THEN P = P - 1 - 3 *
(P = 0 OR P = 3)
380 IF T$ = "{CURSOR RIGHT}" THEN P = P + 3 + 6
(P >2)
```

Logik-Befehle

In Basic gibt es verschiedene Möglichkeiten Vergleiche durchzuführen. Jeden Vergleich können wir auch als Frage ansehen. So zum Beispiel könnte man für den Vergleich »A = B« fragen:

Ist der Wert A ebensogroß wie der Wert B? Als Antwort stände: »das ist wahr« oder »das ist falsch«. Nach diesem Prinzip behandelt auch der Computer die Vergleiche. Wenn man die Antwort jedesmal in einem Satz ausgeben würde, bräuchte man viel Platz. Deshalb hat man sich geeinigt, für den Ausdruck »das ist wahr« (-1) anzugeben und für »das ist falsch« (0). Diese Werte ((-1) und 0) können wir auch zum Rechnen verwenden. In der folgenden Tabelle finden Sie alle Vergleichsbefehle mit ihrer Beschreibung.

=	sind die Werte gleich groß?
<	ist der links stehende Wert kleiner als der rechts stehende?
>	ist der links stehende Wert größer als der rechts stehende?
<>	sind die Werte verschieden groß?
<=	ist der links stehende Wert kleiner oder gleichgroß mit dem rechts stehenden?
>=	ist der links stehende Wert größer oder gleichgroß mit dem rechts stehenden?

Alle in dieser Tabelle stehenden Bedingungen (Vergleiche) können nur mit wahr oder falsch beantwortet werden. Es gibt aber noch eine Reihe anderer Logik-Befehle, wie OR oder AND. Diese unterscheiden sich von den ersten Logik-Befehlen dadurch, daß hier mehrere Antworten möglich sind. Ein Beispiel:

A = (27 AND 8) => A = 8

Warum ist das Ergebnis acht? In unserem Beispiel wird der erste Wert binär mit dem zweiten Wert verglichen.

```
   27 = 0001 1011
AND 8 = 0000 1000
-----------------
    8 = 0000 1000
```

Verknüpfen wir die Bits jeder Spalte, von rechts nach links, nach folgendem Algorithmus: 0 AND 1 oder 1 AND 0 = 0 und 1 AND 1 = 1. Das entspricht also einer einfachen Multiplikation. Das heißt in unserem Beispiel, daß die ersten vier Bit zu Null werden. In der 5. Spalte steht jeweils in der ersten und zweiten Zeile eine 1 und somit ist die Verknüpfung auch eine 1 (1 mal 1 = 1).

Mit dieser Verküpfung können wir bestimmte Werte (binär) aus einem anderen Wert herausfiltern.

Beispiel:

A = (46 AND 15). Hier werden die Wertanteile von 0 bis 15 herausgefiltert.

```
    46 = 0010 1110
AND 15 = 0000 1111
------------------
    14 = 0000 1110
```

Die andere Verknüpfung ist die OR-(Oder-)Verknüpfung. Hier lautet der Algorithmus: ist entweder der eine Wert oder der andere oder beide Werte 1, so ist das Ergebnis 1, sind beide Werte 0, so ist auch das Ergebnis 0.

Ein Beispiel:

```
   13 = 0000 1101
OR 46 = 0010 1110
-----------------
   47 = 0010 1111
```

Mehr über IF-Abfragen und logische Vergleiche finden Sie im Basic-Kurs.

390 IF T$ = "{CURSOR LEFT}" THEN P = P - 3 - 6 (P <3)

Da nur zwei Spalten vorhanden sind, müßte nicht zwischen CURSOR RIGHT und CURSOR LEFT unterschieden werden. Beide Tasten veranlassen nur den Wechsel in die andere Spalte. Eigentlich ließe sich dieser Sonderfall durch eine OR-Verknüpfung einfacher realisieren. Sie lautet:

380 IF T$ = "{CURSOR RIGHT}" OR T$ = "{CURSOR LEFT}" THEN ...

Auf die THEN-Anweisung könnte jede der Formeln aus Zeile 380 oder 390 passen.

Als kleine Beigabe habe ich für Sie noch eine JA/NEIN-Abfrage (Listing 5). Im Prinzip haben Sie hier wieder ein horizontales Menü. Mit den Cursortasten können Sie eine der beiden Antworten anwählen. Durch Drücken der F1- oder RETURN-Taste veranlassen Sie die Ausführung des Unterprogramms für Ja oder Nein. (Florian Müller/do)

```
100 REM ********************************
110 REM *                              *
120 REM * B E I S P I E L - M E N U E *
130 REM *                              *
140 REM ********************************
150 REM *                              *
160 REM *    (SEHR EINFACHES MENUE)    *
170 REM *                              *
180 REM ********************************
190 :
200 L=1: G=1: A=1: REM LEVEL, GESCHWINDIGKEI
    T UND ANZAHL DER SPIELER VORBELEGEN
210 COLOR 0,1: COLOR 4,1: PRINT CHR$(5): SCN
    CLR
220 PRINT TAB(7)"BEISPIELMENUE FUER C16/116"
230 PRINT TAB(7)"--------------------------"
240 PRINT "{4DOWN}"
250 PRINT TAB(3);CHR$(18)" 1 " CHR$(146);TAB
    (10);"SPIELSTUFE{DOWN}"
260 PRINT TAB(3);CHR$(18)" 2 " CHR$(146);TAB
    (10);"GESCHWINDIGKEIT{DOWN}"
270 PRINT TAB(3);CHR$(18)" 3 " CHR$(146);TAB
    (10);"ANZAHL DER SPIELER{DOWN}"
280 PRINT TAB(3);CHR$(18)" 4 " CHR$(146);TAB
    (10);"AUSWAHL BEENDEN{2DOWN}"
290 PRINT "SPIELSTUFE :",L
300 PRINT "GESCHWINDIGKEIT :",G
310 PRINT "ANZAHL DER SPIELER :"A
320 INPUT "{2DOWN}WELCHEN MENUEPUNKT";P
330 IF P<1 OR P>4 OR P<>INT(P) THEN 320: REM
    NUR EINGABEN 1,2,3,4 ZULASSEN
340 :
350 REM UNTERPROGRAMME ANSPRINGEN
360 ON P GOTO 380,450,520,590
370 :
380 REM UNTERPROGRAMM VON MENUEPUNKT 1
390 REM ------------------------------
400 :
410 INPUT "SPIELSTUFE (1,2,3)";L
420 IF L<1 OR L>3 OR L<>INT(L) THEN 410: REM
    NUR RICHTIGE EINGABEN ZULASSEN
430 GOTO 210
440 :
450 REM UNTERPROGRAMM VON MENUEPUNKT 2
460 REM ------------------------------
470 :
480 INPUT "GESCHWINDIGKEIT (1,2,3)";G
490 IF G<1 OR G>3 OR G<>INT(G) THEN 480: REM
    NUR RICHTIGE EINGABEN ZULASSEN
500 GOTO 210
510 :
520 REM UNTERPROGRAMM VON MENUEPUNKT 3
530 REM ------------------------------
540 :
550 INPUT "ANZAHL DER SPIELER (1,2)";A
560 IF A<>1 AND A<>2 THEN 550
570 GOTO 210
580 :
590 REM UNTERPROGRAMM VON MENUEPUNKT 4
600 REM ------------------------------
610 :
620 PRINT "{UP}AUSWAHL UEBER MENUEPUNKT 4 BE
    ENDET.": END
```

Listing 1. Sehr einfaches Menü mit ON n GOTO-Befehl

```
100 REM ********************************
110 REM *                              *
120 REM * B E I S P I E L - M E N U E *
130 REM *                              *
140 REM ********************************
150 REM *                              *
160 REM *     (MIT WINDOW-TECHNIK)     *
170 REM *                              *
180 REM ********************************
190 :
200 L=1: G=1: A=1: REM LEVEL, GESCHWINDIGKEI
    T UND ANZAHL DER SPIELER VORBELEGEN
210 COLOR 0,1: COLOR 4,1: PRINT CHR$(5): SCN
    CLR
220 PRINT TAB(7)"BEISPIELMENUE FUER C16/116"
230 PRINT TAB(7)"--------------------------"
240 PRINT "{5DOWN}"
250 PRINT TAB(3);CHR$(18)" 1 " CHR$(146);TAB
    (10);"SPIELSTUFE{DOWN}"
260 PRINT TAB(3);CHR$(18)" 2 " CHR$(146);TAB
    (10);"GESCHWINDIGKEIT{DOWN}"
270 PRINT TAB(3);CHR$(18)" 3 " CHR$(146);TAB
    (10);"ANZAHL DER SPIELER{DOWN}"
280 PRINT TAB(3);CHR$(18)" 4 " CHR$(146);TAB
    (10);"AUSWAHL BEENDEN{2DOWN}"
290 REM NACH DEN NORMALEN PRINT-ZEILEN  WERD
    EN JETZT DIE WINDOWS GEDRUCKT
300 :
310 REM ZUERST DAS COPYRIGHT-WINDOW:
320 REM ========================
330 :
340 CHAR 1,5,2,"U**********I"
350 CHAR 1,5,3,"=COPYRIGHT="
360 CHAR 1,5,4,"= WINDOW: ="
370 CHAR 1,5,5,"G**********W"
380 CHAR 1,5,6,"=(C) 64'ER="
390 CHAR 1,5,7,"J**********K"
400 :
410 REM NUN DAS LEVEL-WINDOW:
420 REM ===================
430 CHAR 1,22,5,"{RVSON,15SPACE}"
440 CHAR 1,22,6,"{RVSON} SPIELSTUFE:"+STR$(L
    )+" "
450 CHAR 1,22,7,"{RVSON,15SPACE,RVOFF}"
460 :
470 REM UND DAS GESCHWINDIGKEITSWINDOW:
480 REM ============================
490 :
500 CHAR 1,28,10,"R**********S"
510 CHAR 1,28,11,"=TEMPO:"+STR$(G)+" ="
520 CHAR 1,28,12,"Z**********X"
530 :
540 REM UND DAS SPIELERZAHL-WINDOW:
550 REM =========================
560 :
570 CHAR 1,25,16,"*************"
580 CHAR 1,25,17,"*"+STR$(A)+" SPIELER *"
590 CHAR 1,25,18,"*************{DOWN}"+CHR$(
    13)
600 :
610 :
620 PRINT "BITTE ENTSPRECHENDE TASTE DRUECKE
    N !"
630 :
640 GET KEY A$
650 IF A$ = "1" THEN L=L+1: IF L>3 THEN L=1
660 IF A$ = "2" THEN G=G+1: IF G>3 THEN G=1
670 IF A$ = "3" THEN A=A+1: IF A>2 THEN A=1
680 IF A$ = "4" THEN END
690 :
700 GOTO 290
```

Listing 2. Listing mit Window-Technik, einfach gelöst

```
100 REM ******************************
110 REM *                            *
120 REM * M E N U E S T E U E R U N G *
130 REM *                            *
140 REM ******************************
150 REM *                            *
160 REM * VON FLORIAN MUELLER 10/1985 *
170 REM *                            *
180 REM ******************************
190 :
200 REM ***      INITIALISIERUNG     ***
210 REM ***      ===============     ***
220 :
230 CLR
240 A = 5: REM FUENF MENUEPUNKTE
250 DIM R$ (A-1): REM REVERS-TABELLE DIMENSI
    ONIEREN
260 P = 0: REM MENUEPUNKT 0 EINSTELLEN
270 R$ (P) = CHR$(18): REM MENUEPUNKT 0 REVE
    RS
280 COLOR 0,1: COLOR 4,1: PRINT CHR$(142)
290 PRINT CHR$(147): REM BILDSCHIRM LOESCHEN
300 :
310 REM *** AUSGABE DER MENUEPUNKTE ***
320 REM *** ======================= ***
330 :
340 PRINT CHR$(19);: REM CURSOR HOME
350 PRINT "{YELLOW,3SPACE}MENUESTEUERUNG - B
    EISPIELPROGRAMM"
360 PRINT "{3SPACE}=======================
    ========"
370 PRINT : PRINT : PRINT
380 PRINT TAB(9);R$(0);"{RED}MENUEPUNKT A{RV
    OFF} (0){DOWN}"
390 PRINT TAB(9);R$(1);"{CYAN}MENUEPUNKT B{R
    VOFF} (1){DOWN}"
400 PRINT TAB(9);R$(2);"{PURPLE}MENUEPUNKT C
    {RVOFF} (2){DOWN}"
410 PRINT TAB(9);R$(3);"{GREEN}MENUEPUNKT D{
    RVOFF} (3){DOWN}"
420 PRINT TAB(9);R$(4);"{BLUE}MENUEPUNKT E{R
    VOFF} (4){DOWN}"
430 PRINT : PRINT "{GREY3}DERZEIT ANGEWAEHLT
    ER MENUEPUNKT:"P
440 :
450 REM ***     TASTATURABFRAGE      ***
460 REM ***     ===============      ***
470 :
480 GET T$: IF T$="" THEN 480
490 IF T$="{F1}" THEN ON P+1 GOTO 610,680,73
    0,780,830
500 IF T$="{F7}" THEN PRINT "{2DOWN}PROGRAMM
    ABBRUCH.": END
510 R$(P) = CHR$(0)
520 IF T$="{DOWN}" THEN P = P+1 + A*(P=(A-1)
    )
530 IF T$="{UP}" THEN P = P-1 - A*(P=0)
540 IF T$="{HOME}" THEN P = 0
550 R$(P) = CHR$(18)
560 GOTO 310
570 :
580 REM *** MENUEPUNKTE  (ROUTINEN) ***
590 REM *** ======================= ***
600 :
610 REM MENUEPUNKT 1
620 PRINT "{CLR}SIE HABEN DEN OBERSTEN MENUE
    PUNKT ANGE-"
630 PRINT "{DOWN}WAEHLT. DIESER HAT IN DER I
    NTERNEN NU-"
640 PRINT "{DOWN}MERIERUNG DES PROGRAMMS DIE
     NUMMER:"P
650 PRINT "{2DOWN}BITTE DRUECKEN SIE EINE TA
    STE ";
660 POKE 204,0: WAIT 198,1: POKE 204,1: POKE
     198,0: GOTO 290
670 :
680 REM MENUEPUNKT 2
690 PRINT "{CLR}SIE HABEN DEN ZWEITOBERSTEN
    MENUEPUNKT"
700 PRINT "{DOWN}ANGEWAEHLT. INTERNE NUMMER:
     "P
710 GOTO 650
720 :
730 REM MENUEPUNKT 3
740 PRINT "{CLR}SIE HABEN DEN MITTLEREN MENU
    EPUNKT ANGE-"
750 PRINT "WAEHLT. DESSEN INTERNE NUMMER IST
    :"P
760 GOTO 650
770 :
780 REM MENUEPUNKT 4
790 PRINT "{CLR}SIE HABEN DEN ZWEITUNTERSTEN
     MENUEPUNKT"
800 PRINT "{DOWN}ANGEWAEHLT. INTERNE NUMMER:
    "P
810 GOTO 650
820 :
830 REM MENUEPUNKT 5
840 PRINT "{CLR}SIE HABEN DEN UNTERSTEN MENU
    EPUNKT ANGE-"
850 PRINT "WAEHLT. INTERNE NUMMER:"P
860 GOTO 650
```

Listing 3. Vertikal orientiertes Menü.
Vor dem Start unbedingt im Direktmodus eingeben:
KEY 1,CHR$(133):KEY 7,CHR$(136)

```
100 REM ******************************
110 REM *                            *
120 REM * M E N U E S T E U E R U N G *
130 REM *                            *
140 REM ****** GEKUERZTE VERSION ******
150 REM *                            *
160 REM * VON FLORIAN MUELLER 10/1985 *
170 REM *                            *
180 REM ******************************
190 :
200 REM INITIALISIERUNG
210 CLR : DIM R$(4): P=0: R$(P)="{RVSON}"
220 COLOR 0,1: COLOR 4,1: PRINT "{CLR}" CHR$
    (142)
230 REM AUSGABE DER MENUEPUNKTE
240 PRINT "{HOME,YELLOW,3SPACE}MENUESTEUERUN
    G - BEISPIELPROGRAMM"
250 PRINT "{3SPACE}=======================
    ========{2DOWN}"
260 PRINT TAB(11);R$(0);"{RED}MENUEPUNKT A{D
    OWN}"
270 PRINT TAB(11);R$(1);"{CYAN}MENUEPUNKT B{
    DOWN}"
280 PRINT TAB(11);R$(2);"{PURPLE}MENUEPUNKT
    C{DOWN}"
290 PRINT TAB(11);R$(3);"{GREEN}MENUEPUNKT D
    {DOWN}"
300 PRINT TAB(11);R$(4);"{BLUE}MENUEPUNKT E{
    DOWN,GREY3}"
310 REM TASTATURABFRAGE
320 GET T$: IF T$="" THEN 320
330 IF T$="{F1}" THEN ON P+1 GOTO 420,470,50
    0,530,560
340 IF T$="{F7}" THEN PRINT "{2DOWN}PROGRAMM
    ABBRUCH.": END
350 R$(P)=""
360 IF T$="{DOWN}" THEN P=P+1+5*(P=4)
370 IF T$="{UP}" THEN P=P-1-5*(P=0)
380 IF T$="{HOME}" THEN P=0
390 R$(P)="{RVSON}"
400 GOTO 230
410 REM MENUEPUNKTE (ROUTINEN)
420 PRINT "{CLR}SIE HABEN DEN OBERSTEN MENUE
    PUNKT ANGE-"
```

```
430 PRINT "{DOWN}WAEHLT. DIESER HAT IN DER I
    NTERNEN NU-"
440 PRINT "{DOWN}MERIERUNG DES PROGRAMMS DIE
     NUMMER:"P
450 PRINT "{2DOWN}BITTE DRUECKEN SIE EINE TA
    STE ";
460 POKE 204,0: WAIT 198,1: POKE 204,1: POKE
     198,0: GOTO 220
470 PRINT "{CLR}SIE HABEN DEN ZWEITOBERSTEN
    MENUEPUNKT"
480 PRINT "{DOWN}ANGEWAEHLT. INTERNE NUMMER:
    "P
490 GOTO 450
500 PRINT "{CLR}SIE HABEN DEN MITTLEREN MENU
    EPUNKT ANGE-"
510 PRINT "WAEHLT. DESSEN INTERNE NUMMER IST
    :"P
520 GOTO 450
530 PRINT "{CLR}SIE HABEN DEN ZWEITUNTERSTEN
     MENUEPUNKT"
540 PRINT "{DOWN}ANGEWAEHLT. INTERNE NUMMER:
    "P
550 GOTO 450
560 PRINT "{CLR}SIE HABEN DEN UNTERSTEN MENU
    EPUNKT ANGE-"
570 PRINT "WAEHLT. INTERNE NUMMER:"P
580 GOTO 450
```

Listing 4. Verkürztes vertikales Menü.
Vor dem Start unbedingt im Direktmodus eingeben:
KEY 1,CHR$(133):KEY 7,CHR$(136)

```
100 REM *******************************
110 REM *                             *
120 REM * M E N U E S T E U E R U N G *
130 REM *                             *
140 REM ********* HORIZONTAL **********
150 REM *                             *
160 REM * VON FLORIAN MUELLER 10/1985 *
170 REM *                             *
180 REM *******************************
190 :
200 REM INITIALISIERUNG
210 CLR : DIM R$(4): P=0: R$(P)="{RVSON}"
220 COLOR 0,1: COLOR 4,1: PRINT "{CLR}" CHR$
    (142)
230 REM AUSGABE DER MENUEPUNKTE
240 PRINT "{HOME,YELLOW,3SPACE}MENUESTEUERUN
    G - BEISPIELPROGRAMM"
250 PRINT "{3SPACE}=========================
    ========{2DOWN}"
260 PRINT "{DOWN}DIESMAL SIND DIE MENUEPUNKT
    E HORIZONTAL"
270 PRINT "{DOWN}ANGEORDNET.{9DOWN}"
280 PRINT R$(0)"{RED}PUNKT A{RVOFF} "R$(1)"{
    CYAN}PUNKT B{RVOFF} "R$(2)"{PURPLE}PUNKT
     C{RVOFF} ";
290 PRINT R$(3)"{GREEN}PUNKT D{RVOFF} "R$(4)
    "{BLUE}PUNKT E{GREY3}"
300 :
310 REM TASTATURABFRAGE
320 GET T$: IF T$="" THEN 320
330 IF T$="{F1}" THEN PRINT "{DOWN}ANGEWAEHL
    TER PUNKT:"P: END
340 IF T$="{F7}" THEN PRINT "{2DOWN}PROGRAMM
    ABBRUCH.": END
350 R$(P)=""
360 IF T$="{RIGHT}" THEN P=P+1+5*(P=4)
370 IF T$="{LEFT}" THEN P=P-1-5*(P=0)
380 IF T$="{HOME}" THEN P=0
390 R$(P)="{RVSON}"
400 GOTO 230
```

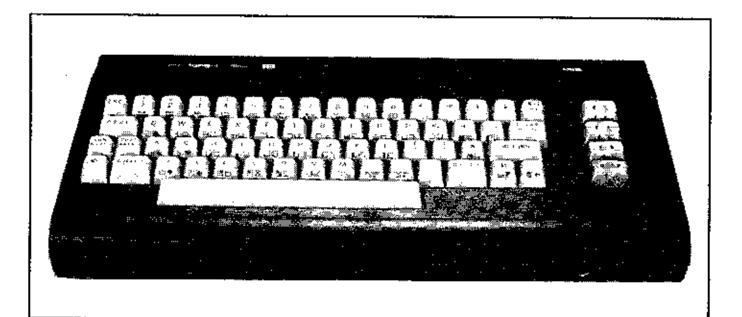
Listing 5. Horizontal orientiertes Menü.
Vor dem Start unbedingt im Direktmodus eingeben:
KEY 1,CHR$(133):KEY 7,CHR$(136)

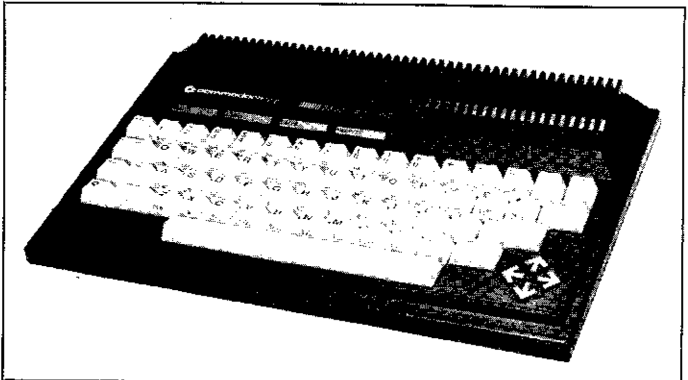
```
100 REM *******************************
110 REM *                             *
120 REM * M E N U E S T E U E R U N G *
130 REM *                             *
140 REM ***** HORIZONTAL/VERTIKAL *****
150 REM *                             *
160 REM * VON FLORIAN MUELLER 10/1985 *
170 REM *                             *
180 REM *******************************
190 :
200 REM INITIALISIERUNG
210 CLR : DIM R$(5): P=0: R$(P)="{RVSON}"
220 COLOR 0,1: COLOR 4,1: PRINT "{CLR}" CHR$
    (142)
230 REM AUSGABE DER MENUEPUNKTE
240 PRINT "{HOME,YELLOW,3SPACE}MENUESTEUERUN
    G - BEISPIELPROGRAMM"
250 PRINT "{3SPACE}=========================
    ========{2DOWN}"
260 PRINT "{DOWN}MENUEPUNKTE SIND HORIZONTAL
     & VERTIKAL"
270 PRINT "{DOWN}ANGEORDNET.{5DOWN}"
280 PRINT TAB(10)R$(0)"{RED}PUNKT A{RVOFF,6S
    PACE}"R$(3)"{GREEN}PUNKT D{DOWN}"
290 PRINT TAB(10)R$(1)"{CYAN}PUNKT B{RVOFF,6
    SPACE}"R$(4)"{BLUE}PUNKT E{DOWN}"
300 PRINT TAB(10)R$(2)"{PURPLE}PUNKT C{RVOFF
    ,6SPACE}"R$(5)"{ORANGE}PUNKT F{GREY3}"
310 REM TASTATURABFRAGE
320 GET T$: IF T$="" THEN 320
330 IF T$="{F1}" THEN PRINT "{DOWN}ANGEWAEHL
    TER PUNKT:"P: END
340 IF T$="{F7}" THEN PRINT "{2DOWN}PROGRAMM
    ABBRUCH.": END
350 R$(P)=""
360 IF T$="{DOWN}" THEN P=P+1+3*(P=2 OR P=5)
370 IF T$="{UP}" THEN P=P-1-3*(P=0 OR P=3)
380 IF T$="{RIGHT}" THEN P=P+3+6*(P>2)
390 IF T$="{LEFT}" THEN P=P-3-6*(P<3)
400 IF T$="{HOME}" THEN P=0
410 R$(P)="{RVSON}"
420 GOTO 230
```

Listing 6. Zweidimensionales Menü.
Vor dem Start unbedingt im Direktmodus eingeben:
KEY 1,CHR$(133):KEY 7,CHR$(136)

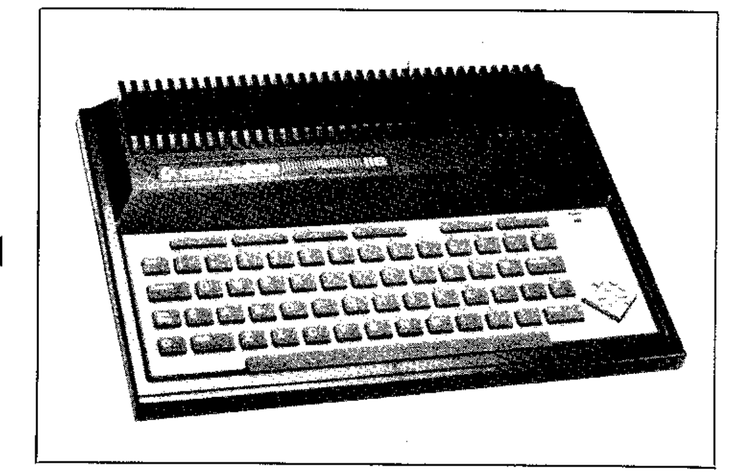
```
100 REM ******************************
110 REM *                            *
120 REM *     SICHERHEITSABFRAGEN    *
130 REM *                            *
140 REM ******************************
150 REM *                            *
160 REM * FLORIAN MUELLER     M-1985 *
170 REM *                            *
180 REM ******************************
190 :
200 CLR : DIM R$(1): P=0: R$(P)="{RVSON}"
210 PRINT "{CLR}SICHERHEITSABFRAGEN (J/N-ENT
    SCHEIDUNGEN)"
220 PRINT "{HOME,6DOWN}WOLLEN SIE EINE ERKLA
    ERUNG? "R$(0)"JA{RVOFF}/"R$(1)"NEIN"
230 GET T$: IF T$="" THEN 230
240 R$(P)=""
250 IF T$="{RIGHT}" OR T$="{LEFT}" OR T$=" "
     THEN P=P+1+2*(P=1)
260 IF T$<>"{F1}" AND T$<>CHR$(13) THEN R$(P
    )="{RVSON}": GOTO 220
270 PRINT "{2DOWN,RVSON}ERGEBNIS:"P"{DOWN}"
280 IF P=0 THEN PRINT "ERKLAERUNG: SIEHE ART
    IKEL"
290 IF P=1 THEN PRINT "DANN EBEN NICHT!"
```

Listing 7. Ja/Nein-Abfrage. Vor dem Start unbedingt im Direktmodus eingeben: KEY 1,CHR$(133):KEY 7,CHR$(136)





Drei »ungleiche« Brüder



Der C16, C116 und Plus/4 sind drei Commodore-kinder, die sich sehr ähnlich sind. Wir sagen Ihnen, wie kompatibel alle drei sind.

Commodore brachte alle drei Computer eigentlich schon 1984 auf den Markt. Doch ein Preis von 1298 Mark für den Plus/4 (oben rechts) schreckte die Käufer zurück. Auch der C 16 (oben links) und C 116 (unten rechts) wurde damals für ein Vielfaches des heutigen Preises verkauft. Seit Ende letzten Jahres erleben der C 16 und C 116 durch die günstigen Angebote einen neuen Boom. Was aber ist mit dem vielgerühmten Plus/4? Er verschwand zwischenzeitig völlig aus den Computergeschäften, um heute wieder aufzublühen – als neuer Einkaufshit für unter 500 Mark, gemeinsam mit dem neuen, schnelleren Floppy-Laufwerk 1551.

Wie Sie sicher wissen, ist der Plus/4 serienmäßig mit 64 KByte RAM und eingebauter Software ausgestattet. Auch den C 16 mit eingebauter 64-KByte-Speichererweiterung bekommt man inzwischen für zirka 200 Mark. Sind all diese Computer untereinander kompatibel, oder gibt es auch hier Schwierigkeiten mit der Software?

Zunächst können wir Ihnen sagen, daß der C 116 völlig mit dem C 16 übereinstimmt. Beide Computer unterscheiden sich nur durch Gehäuse, Tastatur und Platinenlayout. Alle elektronischen Bauteile sind gleich, nur die Aufteilung ist wegen der Größe des C 116 anders. Für die Software sind diese beiden Computer absolut kompatibel. Wenn also in diesem Artikel vom C 16 gesprochen wird, gilt dasselbe auch für den C 116.

Der mit 16 KByte sehr magere Speicher des C 16 läßt sich jedoch bis auf 64 KByte aufrüsten. Hier gibt es allerdings Probleme bei der Kompatibilität nach oben. Einige Programme, die für die Grundversion des C 16 (16 KByte RAM) geschrieben wurden, laufen auf dem aufgerüsteten C 16 und dem Plus/4 nicht.

Beispielsweise, wenn es darum geht, Maschinenprogramme in den erweiterten C 16 oder Plus/4 zu laden. Diese Programme werden manchmal vom Computer nicht ausgeführt. Es liegt daran, daß Maschinenprogramme immer an eine vom Programm bestimmte Speicherstelle geladen werden müssen. Deshalb werden Sie auch »absolut«, das heißt mit der Sekundäradresse 1, geladen. Also »LOAD "Name", 1,1« für Datasette oder »LOAD "Name",8,1« für Diskette. Ohne die Sekundäradresse werden die Programme stets an den Anfang des Basic-Speichers geladen. Aber die wenigsten Maschinenprogramme gehören dorthin.

Kommen wir nun zum eigentlichen Problem. Der Speicher des C 16 in der Grundversion unterscheidet sich nämlich von seinen größeren Brüdern.

Das RAM des C 16 in der Grundversion: $1000 bis $3FFF (dezimal: 4096-16383).

Das RAM des C 16 (erweitert) und Plus/4 : $1000 bis FCFF (dezimal: 4096-64512).

Zunächst sieht es so aus, als würden sich dadurch keinerlei Probleme ergeben. Allerdings ändert sich dies, sobald im erweiterten C 16 die Grafik aufgerufen wird. Der Basic-Anfang verschiebt sich dadurch von $1000 (4096) auf $4000 (16384). Daher laufen einige C 16-Programme auf der erweiterten Version und dem Plus/4 nicht.

Eine softwaremäßige Lösung bietet sich hier an. Begrenzen Sie einfach den Speicher auf das Maß des normalen C 16. Dies können Sie realisieren, indem Sie das Ende des Basic-RAMs heruntersetzen. Die dazu notwendigen Speicherzellen sind 55 und 56. In der erweiterten Version des C 16 steht in Speicherzelle 55 der Wert 0 und in 56 der Wert 243. Beim normalen C 16 steht in 55 der Wert 246 und in 56 steht 63. Also, geben Sie ein:

```
POKE 55, 246 : POKE 56, 63
```

Nach »PRINT FRE(0)« können Sie »12275 BYTES FREE« auf dem Bildschirm lesen, wie bei jedem normalen C 16. Also kein Grund zur Panik, wenn mal ein Programm auf dem Plus/4 oder dem »großen« C 16 nicht laufen sollte!

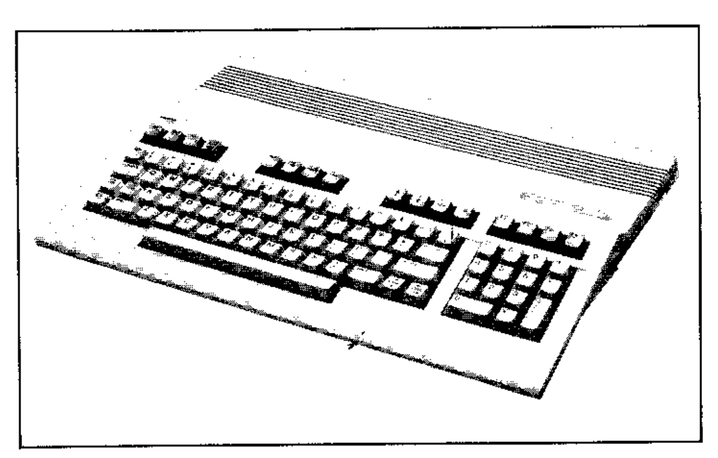
Nach einem Reset werden die Speicherzellen natürlich wieder auf ihre alten Werte gesetzt.

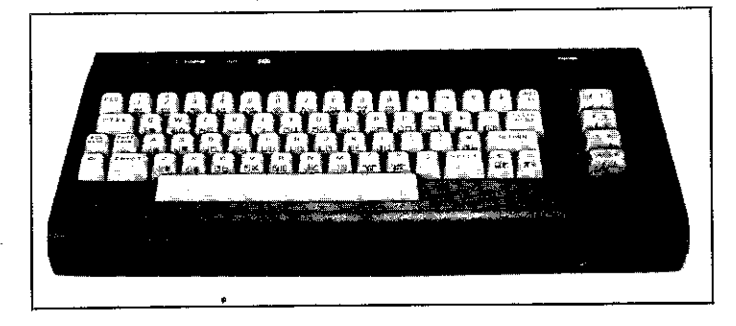
Für den Ablauf von Programmen gibt es zwischen dem Plus/4 und dem auf 64 KByte aufgerüsteten C 16 keine Schwierigkeiten.

(J. Sahlmann/kn)

C 128-Programme auf dem C 16?

Da es für den C16 bisher noch recht wenig Software gibt, werden sich viele C16-Besitzer fragen, wie man ohne großen Aufwand Programme anderer Commodore-Computer umschreiben kann. Die Umwandlung vom Programm für den C128 auf den C16 wollen wir hier etwas näher betrachten.





Im letzten VC20/C16-Sonderheft (SH 3/86) wurden die Gemeinsamkeiten des C 16 und VC 20 behandelt. Dieser Vergleich ermöglichte es bestimmt vielen Lesern, VC 20-Programme an den C 16 anzupassen. Dies ist vor allem möglich, weil diese beiden Computer noch sehr viele Gemeinsamkeiten aufweisen. Eine Anpassung von C 128-Programmen auf den C 16 bedeutet allerdings etwas mehr Aufwand.

Wenn im folgenden Artikel vom C 16 gesprochen wird, ist natürlich ebenso der C 116 und Plus/4 gemeint.

Erste Schwierigkeiten treten auf, wenn man versucht, »einfache« Basic-Programme vom C 128 auf dem C 16 laufen zu lassen. Allzuoft tritt der »beliebte« SYNTAX ERROR auf. Dies liegt daran, daß der C 128 im Vergleich zum C 16 ein noch besseres Basic (Basic 7.0) besitzt. Tabelle 1 enthält den gesamten Befehlsvorrat des C 128. Die hervorgehobenen Befehle sind im Basic 3.5 des C 16 nicht enthalten. Bei einer Anpassung von Basic-Programmen müssen solche Befehle durch geeignete Routinen ersetzt oder gänzlich weggelassen werden.

Zunächst soll versucht werden, die fehlenden Befehle aufzuführen und eine eventuelle Umgehungsmöglichkeit anzubieten.

APPEND

Der Append-Befehl öffnet eine existierende, serielle Datei vom Typ SEQ, USR oder PRG als Ausgabedatei und positioniert den Schreibzeiger auf das Dateiende. Damit besteht die Möglichkeit, zusätzliche Daten in eine schon geschlossene serielle Datei zu schreiben. Der Befehl hat folgendes Format:

APPEND #log Filnr, Dateiname, D Laufw., U Geräteadr.

zum Beispiel:

```
APPEND #2,"TEXT"
```

Dasselbe kann beim C 16 mit dem normalen OPEN-Befehl erreicht werden:

```
OPEN 2,8,2,"TEXT,A"
```

BANK

Die BANK-Anweisung definiert eine von 16 möglichen 64-KByte-Speicherbänken für nachfolgende PEEK-, POKE-, SYS- oder WAIT-Anweisungen. Der BANK-Befehl erübrigt sich beim C 16, da sich seine 16-KByte-RAM-Speicher ohne weiteres über eine 16-Bit-Adreßleitung adressieren lassen.

BEGIN-BEND

Die Anweisungen BEGIN-BEND, welche einen beliebig langen Block von Basic-Anweisungen umschließen, kann man ohne weiteres mit GOTO, GOSUB oder einer IF...THEN-Anweisung umgehen.

BLOAD-BSAVE

Der Befehl BSAVE ermöglicht es, beim C 128 beliebige Speicherbereiche direkt zu speichern. Mit BLOAD kann dieser Datenblock wieder an die gleiche Stelle geladen werden. Die Befehle haben folgendes Format:

BSAVE Dateiname, D Laufwerk, U Gerät, ON B Bank, P Anfangsadresse TO P Endadresse

Sowohl die Anfangs- als auch die Endadresse werden in dezimaler Form erwartet.

BLOAD Dateiname, D Laufwerk, U Gerät, ON B Bank...

Beide Befehle können mit Hilfe einer kleinen Maschinenroutine simuliert werden. Das Programm heißt »BASICTOOL« und ist unter der Rubrik Tips & Tricks zum C 16 nochmals veröffentlicht. Mit Hilfe dieses Programms werden die Befehle BSAVE und BLOAD wie folgt simuliert:

BSAVE: SYS 1548"name",G,1,AA,EA+1

BLOAD: SYS 1536"name",G,1

Hierbei gibt G die Geräteadresse an, AA steht für die Anfangsadresse und EA für die Endadresse.

BOOT

Der Befehl BOOT erübrigt sich beim C 16, da er nicht CP/M-fähig ist und deshalb auch kein CP/M-Programm von Diskette laden und starten kann.

BUMP

Da der C 16 keine Sprites besitzt, erübrigt sich auch ein Ersatz für die BUMP-Funktion des C 128. Dieser Befehl liefert die Sprite-Nummer bei einer Sprite-Kollision.

CATALOG-DIRECTORY

Der CATALOG-Befehl listet das Directory der Diskette auf. Gleiches bewirkt auch der DIRECTORY-Befehl des C 16 und C 128. Warum man beim C 128 für ein und dieselbe Funktion zwei Befehle benötigt, ist mir ein Rätsel, zumal beide Befehle dieselbe Betriebssystemroutine verwenden.

COLLISION

Auch die COLLISION-Anweisung für die Sprites des C 128 ist auf dem C 16 nicht zu ersetzen.

CONCAT

Der CONCAT-Befehl des C 128 hängt an eine sequentielle

Datei eine andere gleichartige Datei an. Das heißt, es können auf einer Diskette mehrere Dateien des Typs SEQ zusammengefaßt werden. Dabei wird die Zieldatei, also die Datei, an die eine andere angehängt wird, verändert. Sie existiert nach dem Kommando in ihrer ursprünglichen Form nicht mehr. Die angehängte Datei bleibt dagegen weiterhin zusätzlich in der alten Form im Directory.

Zum Beispiel:

```
CONCAT "DATEN1" TO "DATEN2"
```

Hier wird die Datei DATEN1 an DATEN2 angehängt. DATEN2 wird dadurch verändert.

Beim C 16 kann man den CONCAT-Befehl mit der COPY-Anweisung über den Kommandokanal der Floppy umgehen:

```
OPEN 1,8,15
PRINT#1,"C0:DATEN2=0:DATEN2,0:DATEN1
CLOSE 1
```

Auch hier wird die Datei DATEN1 an DATEN2 angehängt. DATEN2 wird dadurch verändert. Will man beim C 16 eine Veränderung der Datei DATEN2 verhindern, aber trotzdem beide Dateien verknüpfen, verwendet man folgende Anweisung:

```
OPEN 1,8,15
PRINT#1,"C0:DATEN3=0:DATEN2,0:DATEN1
CLOSE 1
```

Eine Kopie der Dateien DATEN2 und DATEN1 wird zur Datei DATEN3 zusammengefaßt. Die Ursprungsdateien bleiben in ihrer alten Form unverändert.

DCLEAR

Der Befehl DCLEAR zum Schließen aller geöffneten Diskettenkanäle für ein bestimmtes Laufwerk muß im Basic 3.5 durch den Befehl CLOSE umgangen werden.

DCLOSE-DOPEN

Auch der Befehl DCLOSE zum Schließen aller Dateien auf einer Diskette muß durch CLOSE ersetzt werden.

Zur Eröffnung einer Datei auf Diskette steht beim C 128 der DOPEN-Befehl zur Verfügung. So kann mit DOPEN1,...Kanal 1 zum Floppy eröffnet werden. Beim C 16 wird dasselbe durch OPEN 1,8,... erreicht.

DVERIFY

Zusätzlich zu den Befehlen DSAVE und DLOAD zum Speichern und Laden von Programmen auf Diskette wurde beim C 128 der DVERIFY-Befehl integriert. Während das Laden und Speichern mittels DLOAD und DSAVE beim C 16 möglich ist, muß die Programmspeicherung auf Disk mit VERIFY "PROGRAMM",8 (eventuell auch noch »,1«) überprüft werden.

ENVELOPE-FILTER

Da der C 16 leider keinen SID-Chip für die Tonerzeugung hat, erübrigt sich auch der ENVELOPE-Befehl zur Definition einer Hüllkurve.

Der Tongenerator des C 16 ist nur über die Befehle VOL und SOUND ansteuerbar. Alle anderen beliebten Synthesizerfunktionen, die man vom C 64 oder C 128 kennt, sind beim C 16 nicht integriert. Deshalb hat auch die FILTER-Anweisung, zum Setzen der Klangfilterparameter des SID, beim C 16 keine »Daseinsberechtigung«.

FAST-SLOW

Weil man den C 16 im Gegensatz zum C 128 nicht auf doppelte Geschwindigkeit schalten kann, sind die Befehle FAST und SLOW, zum Wählen der Taktfrequenz zwischen 1 und 2 MHz, nicht sinnvoll.

FETCH - STASH - SWAP - GO64

Die Befehle FETCH, STASH und SWAP zur RAM-Floppy-Verwaltung sind auch beim C 128 nur mit eingesteckter RAM-Floppy-Karte wirksam. Da mir nicht bekannt ist, daß von Commodore eine RAM-Floppy für den C 16 angeboten wird, sind diese Befehle beim C 16 unnötig.

Mit der Anweisung GO64 kann man den C 128 zum C 64 umschalten. Da aber der C 16 nicht den Luxus von mehreren Computern in einem Gehäuse bietet, erübrigt sich auch dieser Befehl beim C 16.

GRAPHIC-MOVSPR

Der GRAPHIC-Befehl zum Umschalten in die Grafik-Modi existiert bei beiden Geräten. Der C 128 berücksichtigt zusätzlich die 80-Zeichen-Darstellung.

Die MOVESPR-Anweisung zur Sprite-Animation entfällt beim C 16 wegen »akutem Sprite-Mangel«.

PEN-POT

Die Befehle PEN zur Abfrage des Lichtgriffels und POT zur Überprüfung des Paddles sind beim C 16 nicht vorhanden und müssen bei Bedarf durch selbstgeschriebene Programme ersetzt werden.

PLAY

Die PLAY-Anweisung zum Spielen einer Tonsequenz läßt sich beim C 16 sehr leicht durch ein kleines Basic-Programm ersetzen, zumal sich der simple Tongenerator mit Tonhöhe, Notenlänge und Lautstärke zufriedengibt.

POINTER

Der POINTER-Befehl gibt beim C 128 die Adresse einer Variablen im Speicher an. Diese nützliche Funktion fehlt leider beim C 16 und läßt sich auch nicht ohne weiteres durch ein »kleines« Basic-Programm ersetzen. Für einen geübten Programmierer dürfte aber eine entsprechende Maschinenroutine kein großes Problem darstellen.

RECORD

Zum komfortablen Positionieren des Schreib/Lese-Zeigers innerhalb einer REL-Datei, auf Diskette, steht dem C 128 Benutzer der RECORD-Befehl zur Verfügung. Der RECORD-Befehl hat folgenden Syntax:

```
RECORD #LA,R,B
```

Hierbei gibt #LA die logische Adresse der Datei, R die Record-Nummer und B die Stelle im Record an. Die Anweisung RECORD #1,200 positioniert zum Beispiel den Record-Zeiger der Datei mit LA=1 auf 200.

Will man beim C 16 den Schreib/Lese-Zeiger positionieren, muß man sich des Befehls POSITION im Kommandokanal der Floppy bedienen. Dieser hat folgendes Format:

P SA RL RH B

Hier gibt SA die Sekundäradresse für die Position an. Zu diesem Wert muß 96 addiert werden. Wenn Sie die Datei durch OPEN 1,8,2,..., also mit SA=2 geöffnet haben, so müssen Sie beim POSITION-Befehl für SA den WERT 98 (2+96) angeben.

RL und RH beinhalten die Record-Nummer zerlegt in Low- und High-Byte. Diese ist nötig, weil mit einem Byte nur 256 Werte dargestellt werden können, aber eine relative Datei mehr als 256 Records haben kann. Wenn R die absolute Record-Nummer ist, errechnen sich beide Parameter wie folgt:

RH = INT (R/256)
RL = R-RH*256

Der letzte Parameter stimmt mit der entsprechenden Angabe im RECORD-Befehl beim C 128 überein. Er gibt auch hier die Stellung im Record an. Dieser Parameter kann in beiden Fällen nur Werte von 1 bis 254 annehmen. Alle vier Parameter sind BYTE-Parameter und müssen mit Hilfe der CHR$()-Anweisung gesendet werden.

Will man beim C 16 beispielsweise eine REL-Datei mit der Kanalnummer 5 eröffnen und den Schreib/Lese-Zeiger auf Record 1000, Byte 15 stellen, so benutzt man folgende Anweisungen:

```
OPEN 1,8,15
RH = INT (1000/256)
RL = 1000-RH*256
PRINT#1,"P" + CHR$(5+96) + CHR$(RL) + CHR$(RH)
+ CHR$(15)
CLOSE 1
```

RREG

Die RREG-Anweisung des C128 weist vier Variablen die Inhalte der Prozessorregister (Akku, X-Reg., Y-Reg.) und des Statusregisters zu. Beim C16 lassen sich diese Werte durch die altbekannte PEEK-Anweisung auslesen:

A-Reg. = PEEK (2034)
X-Reg. = PEEK (2035)
Y-Reg. = PEEK (2036)
Statusreg. = PEEK (2037)

RSPRCOLOR - RSPPOS - PSPRITE

Folgende Befehle zur Sprite-Animation und Kontrolle sind, wie schon erwähnt, beim C16 nicht zu realisieren. Zur Sprite-Abfrage dienen beim C128 die Befehle RSPRCOLOR, RSPPOS und PSPRITE. Zur Sprite-Definiton und -Animation können die Basic-Befehle SPRCOLOR, SPRDEF, SPRITE und SPRSAV verwendet werden.

WINDOW - RWINDOW

Während der C128 fähig ist, mehrere Windows zu definieren, steht bei dem C16 nur ein Fenster zur Verfügung. Dieses kann zudem nur über die ESC-Taste definiert werden. Der C128 bietet dagegen komfortable Window-Befehle. Mit der Anweisung WINDOW lassen sich die Bildschirmfenster für die Textausgabe setzen.

Will man die aktuellen Parameter eines Windows abfragen, so steht dem C128-Nutzer der Befehl RWINDOW zur Verfügung. Damit lassen sich allerdings nur drei Parameter abfragen:
- letzte Zeile des Fensters
- letzte Spalte des Fensters
- Bildschirmbreite (40 oder 80 Zeichen)

Beim C16 ist der Aufwand, im Direktmodus ein Window zu definieren, keineswegs hinderlich, sondern bringt eher Vorteile, da kein Befehl »Window« am Bildschirm erscheint. Im laufenden Programm ist es jedoch sehr umständlich, ein Fenster zu generieren. Es sind sehr viele PRINT-Anweisungen notwendig.

Da die WINDOW-Funktion zweifellos in vielen C16-Programmen nützlich wäre, lohnt es sich schon, diesen Befehl im C16-Wortschatz aufzunehmen. Ein kleines Maschinenprogramm aus dem 64'er-Sonderheft 3/86, Seite 174, macht es möglich. Nach dem Initialisieren des Programms steht dem C16-Nutzer der Befehl FENSTER zur Verfügung. Mit ihm läßt sich durch Angabe von vier Parametern ein Window eröffnen. Weitere Anregungen zur Window-Technik auf dem C16 sind den Artikeln »Das Auge ißt mit« in dieser Ausgabe und »Windows - Fenster zum neuen Bedienungskomfort« im Sonderheft 7/86 zu entnehmen.

Um die Parameter des Bildschirmfensters zu erfahren bedient man sich ebenfalls der PEEK-Anweisung. In den Systemadressen lassen sich unter den Labels SCBOT, SCTOP, SCLF und SCRT die Bildschirmgrenzen abrufen:

BESCHREIBUNG	LABEL	ADRESSE
Unterer Bildrand:	SCBOT	PEEK(2021)
Oberer Bildrand:	SCTOP	PEEK(2022)
Linker Bildrand:	SCLF	PEEK(2023)
Rechter Bildrand:	SCRT	PEEK(2024)

SLEEP

Um den C16 in der Programmabarbeitung für eine gewisse Zeit anzuhalten, bedient man sich in der Regel einer FOR...NEXT-Schleife. Beim C128 kann der Computer mit SLEEP für eine bestimmte Zeit (zwischen 1 und 65535 Sekunden) »eingefroren« werden. Will man den C16 ebenfalls für eine ganz bestimmte Zeit stillegen, so kann man sich der Variablen der Systemuhr (TI und TI$) bedienen.

TEMPO

Der TEMPO-Befehl setzt das Abspieltempo für die PLAY-Anweisung beim C128. Da der C16 aber keinen PLAY-Befehl besitzt, hat auch TEMPO wenig Sinn.

WIDTH

Mit Ausnahme einer Anweisung besitzen beide Computer dieselben Grafik-Befehle. Um die Strichstärke bei allen Grafik-Befehlen zu variieren, besitzt der C128 zusätzlich den Befehl WIDTH. Mit ihm kann zwischen einfacher und doppelter Strichstärke umgeschaltet werden. Soll beim C16 doppelt-dick gezeichnet werden, muß wohl oder übel der Grafik-Befehl zweifach angewandt werden. Zwischen beiden Anweisungen sollte man natürlich die entsprechenden Parameter um jeweils ein Pixel vergrößern oder verkleinern.

XOR

Als letzter Basic-Befehl ist im Basic 7.0 zusätzlich zu den anderen logischen Operationen (and, or, not) die XOR-Anweisung hinzugekommen. Diese dient zur Exklusiv-Oder-Verknüpfung zweier Werte. Einen entsprechenden Ersatz gibt es dafür beim C16 leider nicht. Die einzige Möglichkeit wäre, die Werte in 16-Bit-Zahlen zu zerlegen und diese »von Hand« exklusiv-oder-verknüpfen. So ergibt die Anweisung: PRINT XOR (124,12) den Wert 112. Von Hand sieht die Sache so aus:

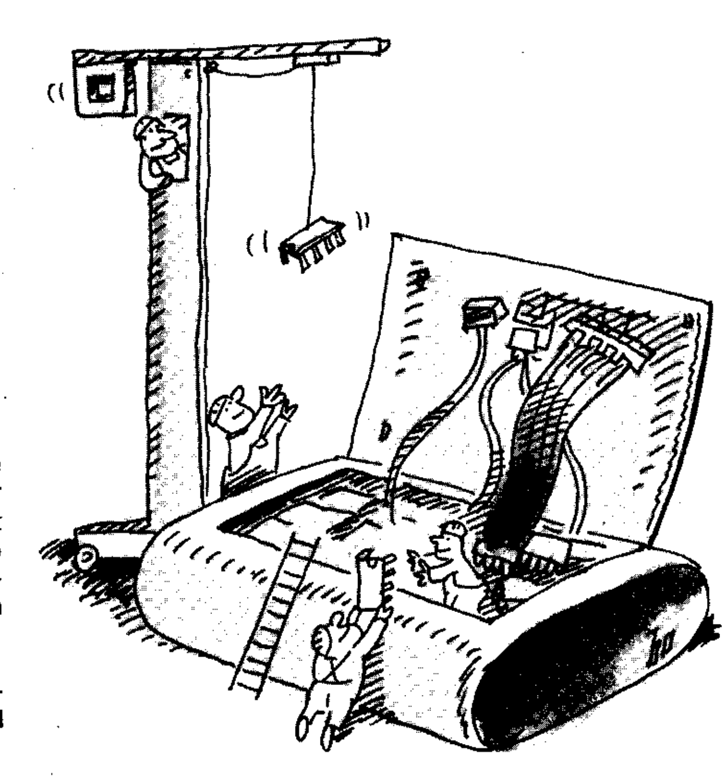
Dezimalzahl	Binärschreibweise:
124	0000000001111100
12	0000000000001100
XOR 112	0000000001110000

Ein entsprechendes Basic-Programm zur Umwandlung von Dezimalzahlen in Binärschreibweise und anschließende XOR-Verknüpfung wäre relativ lang und würde den Rahmen dieses Artikels sprengen.

Maschinenprogramme

Soweit zur Basic-Anpassung der C128-Programme an den C16. Bis zu diesem Punkt ist es für den C16-Anwender noch recht einfach, Programme für sein System umzusetzen. Wenn es darum geht, Maschinenprogramme umzuschreiben, wird die Sache schon komplizierter.

Zunächst werfen wir einen Blick auf die Prozessoren beider Geräte. Der C16 sowie der C128 besitzen sozusagen einen 65 xx-Prozessor. Im C16 arbeitet ein Prozessor mit der



Bezeichnung 7501. Diese Commodore-Entwicklung hat denselben Befehlsvorrat wie der 6502 des VC 20 oder 6510 des C 64. Er unterscheidet sich unter anderem dadurch, daß er das Bankswitching unterstützt. Diese Option erscheint für den C 16/116 überflüssig, da er mit seinem 16-KByte-RAM-Speicher keine Probleme haben dürfte, Betriebssystem, Basic-Interpreter und RAM-Speicher mit einer 16-Bit-Adreßleitung zu adressieren. Da aber auch der Plus/4 mit eingebauter Software und 64-KByte-RAM-Speicher denselben Prozessor besitzt, ist die Option BANKSWITCHING durchaus sinnvoll. Wenn Sie nun glauben, daß diese Funktion beim C 16 zwar vorhanden ist, aber nicht genutzt wird, so haben Sie weit gefehlt. Ein Beweis hierfür ist eine 64-KByte-Karte für den C 16. Diese stellt dem Benutzer wie beim Plus/4 fast 64-KByte-Basic-Speicher zur Verfügung. Das »Herz« des C 128, der 8502-Prozessor, ist ebenfalls eine Entwicklung von Commodore. Er wurde speziell für den C 128 entwickelt und enthält ebenfalls alle Eigenschaften der Vorgänger. Dieser Prozessor hat den Vorteil, daß er im Gegensatz zu seinen Vorgängern mit unterschiedlicher Frequenz getaktet werden kann (erinnern wir uns an die Basic-Befehle FAST und SLOW).

Ein wichtiger Teil für Maschinen-Programmierer ist die Zeropage bei beiden Computern. Dieser Bereich ist einer der wichtigsten Bereiche des Computers. Hier werden die meisten Werte von ROM-Operationen »zwischengelagert«. Außerdem liegen hier viele Betriebssystem-Vektoren. Da der Zeropage-Bereich im RAM-Bereich liegt, kann man durch Veränderung von Vektoren und Parametern fast beliebig ins »Computergeschehen« eingreifen. Aber Vorsicht: Unkontrollierte Änderungen (durch POKE) im Zeropage führen fast immer zu einem Totalabsturz!

Die Zeropage beider Computer ist ähnlich aufgebaut. Dies können sie den Tabellen 2 und 3 entnehmen. Natürlich verfügt der C 16 nicht über alle Funktionen des C 128. Im folgenden sollen die Zeropage-Adressen mit gleicher Bedeutung gegenübergestellt werden.

Die Adressen 0 und 1 haben bei beiden Computern die gleiche Funktion. Folgende Speicherstellen des C 128 (Adresse 9 bis 98) liegen beim C 16 zwei Byte tiefer (also: Adresse 7 bis 96) und haben im großen und ganzen dieselben Aufgaben. Einige Bytes werden beim C 128 mit einer Doppelfunktion belegt, die aber beim C 16 keine Rolle spielen. Auch die folgenden Speicherstellen 99 bis 105 und 106 bis 111 enthalten dasselbe, wie die C 16-Speicherstellen 97 bis 103 und 105 bis 110. Der Überlauf für den Fließkomma-Akkumulator 1 fehlt beim C 128 an der entsprechenden Stelle. Das heißt, daß im folgenden der Abstand zwischen den Speicherstellen des C 16 und C 128 auf ein Byte zusammenschrumpft: Die Zeropage-Adressen von 112 bis 126 befinden sich beim C 16 zwischen 111 und 127. Wo sich beim C 16 der Arbeitsbereich für den Sound befindet (126 bis 143), wurden beim C 128 nützliche Parameter für die Grafik eingeschoben.

Identisch sind die Adressen 144 bis 150. Der nun folgende Bereich der Zeropage (151 bis 255) enthält im großen und ganzen dieselben Parameter und Zeiger. Ihre Lage wurde aber dermaßen »durcheinandergewürfelt«, daß eine genaue Gegenüberstellung sehr umfangreich ausfallen würde. Aus den Tabellen 2 und 3 kann aber nach Bedarf die gesuchte Adresse sehr leicht entnommen werden.

Die Zeiten, in denen man die wichtigsten Systemadressen innerhalb der ersten 256 Byte unterbrachte, sind jedoch längst vorbei. Nur die alten Commodore-Geräte (PET, CBM 30xx und CBM 40xx) kamen mit der Zeropage aus. Die Nachfolger benötigten immer mehr Systemadressen für Farbe, Grafik, Sprites und so weiter. Beim C 16 wurden schließlich acht »Seiten« mit jeweils 256 Byte verwendet (Adresse 0 bis 2047) (Tabelle 4).

Der C 128 mußte bedingt durch den 40-Zeichen-Bildschirm im Bereich 1024 bis 2023 die Systemadressen bis zur Adresse 4860 ausweiten (Tabelle 5).

All diese Systemadressen zu erläutern und zu vergleichen würde dieses Sonderheft füllen. Um Ihnen einen Überblick zu ermöglichen, werden die erweiterten Zeropages in den Tabellen 4 und 5 gegenübergestellt.

Zum Schluß noch ein weiterer wichtiger Punkt für Maschinen-Programmierer. Viele Maschinenprogramme nutzen die sogenannten Kernel-Sprungadressen am Ende des Betriebssystems für vielfältige Aufgaben. Hier sind Betriebssystemroutinen adressiert, die sehr häufig benötigt werden. Die wohl bekannteste ist die »BSOUT«-Routine bei $FFD2 (Dez: 56490), die ein Zeichen am Bildschirm ausgibt. Auch beim C 128 wurden diese Sprungadressen implementiert. Das heißt, Sprünge ins Betriebssystem ab Adresse $FF81 (Dez: 56490) müssen nicht angepaßt werden.

Fazit:

Beide Computer haben mehr Gemeinsamkeiten, als zunächst angenommen. Obwohl eine Anpassung der C 128-Programme etwas aufwendiger ist als beim VC 20, ist es doch machbar. Ich hoffe, Sie haben mit diesem Artikel ein Mittel in Händen, das Ihnen hilft, diese »schwierige« Arbeit zu erleichtern. (Christian Quirin Spitzner/bj)

Befehl	Funktion
ABS	Liefert absoluten Wert eines Ausdrucks
AND	Logische Verknüpfung
APPEND	Öffnet eine sequentielle Datei zur Datenanfügung
ASC	Liefert ASCII Wert eines Zeichens
ATN	Liefert Arcus Tangens
AUTO	Automatische Zeilennumerierung
BACKUP	Kopiert eine komplette Diskette
BANK	Wählt Speicherbank für PEEK, POKE und SYS
BEGIN-BEND	Faßt mehrere Basic-Zeilen zu einem Block zusammen
BLOAD	Lädt beliebigen Speicherbereich von Floppy
BOOT	Lädt und startet CP/M von Diskette
BOX	Zeichnet Rechteck
BSAVE	Speichert beliebige Speicherbereiche auf Floppy
BUMP	Liefert die Spritenummer bei Sprite-Kollisionen
CATALOG	Listet Inhaltsverzeichnis der Diskette
CHAR	Fügt Text in hochauflösende Grafik ein
CHR$	Liefert Zeichenkette
CIRCLE	Zeichnet Kreise, Ellipsen und Vielecke
CLOSE	Schließt einen Ein-/Ausgabe-Kanal
CLR	Löscht alle Variablen
CMD	Adressiert ein Gerät an einer Ein-/Ausgabe-Schnittstelle
COLLECT	Löscht offene Dateien und reorganisiert Diskette
COLLISION	Dient zur Sprite-Kollisions-Abfrage
COLOR	Setzt Farben für Text und Grafik
CONCAT	Verbindet zwei sequentielle Dateien miteinander
CONT	Setzt Basic-Programm, das durch Drücken der STOP-Taste beendet wurde, fort
COPY	Kopiert Disketten-Dateien
COS	Liefert den Cosinus
DATA	Speichert Daten im Basic-Programm für READ-Anweisung
DCLEAR	Schließt alle Kanäle zur Diskettenstation
DCLOSE	Schließt Kanal zur Diskettenstation
DEC	Liefert Dezimalwert einer Hexadezimalzahl
DEF	Definiert und benennt eine Funktion
DELETE	Löscht einen Zeilenbereich aus einem Programm
DIM	Definiert und reserviert die maximale Anzahl von Elementen einer Feld-Variablen
DIRECTORY	Listet Inhaltsverzeichnis der Diskette
DLOAD	Lädt ein Programm von Diskette
DO...LOOP	Programmschleife LOOP springt immer zu DO zurück
DOPEN	Öffnet Kanal zur Diskettenstation
DRAW	Setzt Punkte oder zeichnet Linien
DS	Enthält den Fehlerstatus des Diskettenlaufwerks

Tabelle 1. Befehlsvorrat des C 16 und C 128. Die hervorgehobenen Befehle existieren nur im Basic 7.0 des C 128.

DS$	Enthält Fehlerstatus der Floppy im Klartext
DSAVE	Speichert ein Programm auf Diskette
DVERIFY	Überprüft Programmspeicherung auf Diskette
EL	Enthält Zeilennummer bei Auftreten eines Fehlers
ELSE	Alternative bei IF-THEN, falls Bedingung nicht erfüllt
END	Beendet Programm
ENVELOPE	Definiert Hüllkurve für Synthesizer
ER	Liefert den Code des zuletzt aufgetretenen Fehlers
ERR$	Liefert Fehlermeldung im Klartext
EXIT	Dient zum Verlassen einer DO...LOOP-Schleife
EXP	Liefert Potenz der Zahl e
FAST	Schaltet auf doppelte Geschwindigkeit
FETCH	Holt Daten aus beliebiger Speicherbank (RAM-Floppy)
FILTER	Setzt den Klangfilterparameter für den SID
FNxx	Bearbeitet Funktion xx
FOR..NEXT	Programmschleife mit einer definierten Zahl von Durchläufen
FRE	Liefert freien Speicherplatz
GET	Holt ein Zeichen von der Tastatur
GET#	Liest Zeichen aus einer Datei
GETKEY	Wartet auf Tastendruck und holt Zeichen
GO64	Schaltet in den C64-Modus
GOSUB	Verzweigt in ein Unterprogramm
GOTO	Verzweigt zu einer bestimmten Zeile
GRAPHIC	Wählt Grafik-Modus
GSHAPE	Schreibt ein SHAPE aus einem String auf den Bildschirm
HEADER	Dient zur Formatierung von Disketten
HELP	Listet nach Fehlermeldung die Fehlerzeile am Bildschirm auf
HEX$	Wandelt Dezimalzahl in Hexadezimal-Strings
IF	Stellt Bedingungen und verzweigt im Programm
INPUT	Erlaubt Dateneingabe über die Tastatur
INPUT#	Liest Daten aus einer sequentiellen oder relativen Datei
INSTR	Ergibt Position eines Teilstrings in einem anderen String
INT	Rundet auf eine ganze Zahl ab
JOY	Fragt Joystick-Position ab
KEY	Dient zur Belegung der Funktionstasten
LEFT$	Holt den linken Teil einer Zeichenkette
LEN	Ermittelt die Länge einer Zeichenkette
LET	Weist einer Variablen einen Wert zu (LET ist nicht zwingend)
LIST	Listet Programm auf
LOAD	Lädt ein Programm
LOCATE	Positioniert den Grafikcursor
LOG	Liefert den natürlichen Logarithmus
DO...LOOP	Programmschleife LOOP springt immer zu DO zurück
MID$	Holt einen Teilstring mitten aus einer anderen Zeichenkette
MONITOR	Ruft den eingebauten Maschinensprache-Monitor auf (TEDMON)
MOVESPR	Bewegt einen Sprite über den Bildschirm
NEW	Löscht Programm und alle Variablen
NOT	Logische Verknüpfung
ON..GOSUB	Verzweigt zu einem von mehreren Unterprogrammen in Abhängigkeit einer Variablen
ON...GOTO	Verzweigt zu einer von mehreren spezifizierten Zeilennummern in Abhängigkeit einer Variablen
OPEN	Eröffnet einen Ein-/Ausgabe-Kanal oder eine Datei
OR	Logische Verknüpfung
PAINT	Füllt einen Bereich der hochauflösenden Grafik aus
PEEK	Liefert den Inhalt einer Speicherstelle
PEN	Fragt Lightpen ab
PLAY	Spielt die in einem String abgelegte Tonfolge
POINTER	Ergibt die Adresse einer Variablen im Speicher
POKE	Schreibt einen Wert zwischen 0 und 255 in eine Speicherstelle
POS	Liefert die Spaltenposition des Cursors
POT	Fragt Paddles ab
PRINT	Gibt Daten am Bildschirm aus
PRINT#	Gibt Daten über einen spezifizierten Ausgabekanal aus
PUDEF	Definiert Steuerzeichen für PRINT USING
RCLR	Liefert gewählten Farbcode für Text und Grafik
RDOT	Liefert Bildschirmposition des Grafik-Cursors
READ	Liest Daten aus DATA-Anweisung
RECORD	Positioniert Schreib-/Lesezeiger bei relativen Dateien
REM	Dient zur Dokumentation eines Programms
RENAME	Dient zum Umbenennen von Diskettendateien
RENUMBER	Numeriert das Basic-Programm neu
RESTORE	Setzt DATA-Zeiger auf beliebige Zeilennummer
RESUME	Rückkehr aus einer Fehlerbehandlungsroutine

Tabelle 1. Befehlsvorrat des C16 und C128. Die hervorgehobenen Befehle existieren nur im Basic 7.0 des C128 (Fortsetz.).

RETURN	Kehrt aus Unterprogramm zurück
RGR	Liefert die Nummer des eingestellten Grafikmodus
RIGHT$	Holt den rechten Teil einer Zeichenkette
RLUM	Gibt die der Farbzone zugewiesene Farbintensität an
RND	Liefert eine Zufallszahl zwischen 0 und 1
RREG	Weist Variablen die Werte der Prozessorregister zu
RSPRCOLOR	Liefert den aktuellen Code des Mehrfarbenmodus für Sprites
RSPPOS	Liefert Position und Geschwindigkeit eines Sprites
RSPRITE	Ergibt je nach Parameter alle Sprite-Attribute
RUN	Startet ein Basic-Programm
RWINDOW	Liefert Parameter des eingestellten Bildschirmfensters
SAVE	Speichert ein Basic-Programm
SCALE	Maßstabswahl bei hochauflösender Grafik
SCNCLR	Löscht Text- oder Grafikbildschirm
SCRATCH	Löscht eine Diskettendatei
SGN	Liefert Vorzeichen eines Argumentes
SIN	Liefert Sinus
SLEEP	Hält die Programmausführung für eine wählbare Zeit an
SLOW	Schaltet auf 1 MHz Takt
SOUND	Erzeugt Toneffekte mit wählbarer Frequenz und Dauer
SPC	Liefert eine bestimmte Anzahl von Leerzeichen (Spaces) in der PRINT-Anweisung
SPRCOLOR	Setzt im Mehrfarben-Modus Farben für Sprites
SPRDEF	Ruft den integrierten Sprite-Editor auf
SPRITE	Setzt Sprite-Attribute
SPRSAV	Speichert ein Sprite in einem String oder umgekehrt
SQR	Liefert Quadratwurzel
SSHAPE	Speichert ein Shape in eine Stringvariable
ST	Liefert Statusbyte
STASH	Überträgt Daten in eine Speicherbank (RAM-Floppy)
STEP	Gibt Schrittweite bei FOR...NEXT-Schleife
STOP	Bricht Programm ab
STR$	Liefert Zeichenkettendarstellung einer Zahl
SWAP	Tauscht Daten zwischen zwei Speicherbänken aus
SYS	Maschinensprache-Aufruf
TAB	Tabuliert den Cursor
TAN	Liefert Tangens
TEMPO	Setzt Abspieltempo für PLAY-Anweisung
THEN	Führt Bedingung nach IF-Anweisung aus
TI	Liefert Stand der internen Systemuhr
TI$	Uhrzeit im Klartext (HHMMSS für Stunden, Minuten, Sekunden)
TO	Gibt das Endziel einer FOR...NEXT-Schleife an (FOR X=1 TO 1000...)
TRAP	Verzweigt im Fehlerfall zu einer Fehlerbehandlungsroutine
TROFF	Schaltet Programmablaufverfolgung (Trace) aus
TRON	Schaltet Trace ein
UNTIL	Setzt Bedingung für DO...LOOP fest (DO UNTIL...)
USING	Erlaubt formatierte Zahlenausgabe
USR	Hilfsmittel zur Übergabe und Bearbeitung einer Variablen an ein Maschinenprogramm
VAL	Liefert den numerischen Wert einer Zeichenkette, die ausschließlich aus Zahlen besteht
VERIFY	Überprüft Programmspeicherung
VOL	Setzt Lautstärke für die SOUND-Anweisung
WAIT	Wartet, bis eine Speicherstelle ein spezifiziertes Bitmuster angenommen hat
WHILE	Setzt Bedingung für DO...LOOP fest (DO...WHILE...)
WIDTH	Setzt die Strichstärke für alle Grafikbefehle
WINDOW	Definiert ein Bildschirmfenster
XOR	Liefert die Exklusiv-Oder-Verknüpfung zweier Werte

Tabelle 1. Befehlsvorrat des C16 und C128. Die hervorgehobenen Befehle existieren nur im Basic 7.0 des C128 (Schluß).

Dezimal	Hexadezimal	Bemerkung
0	$0000	Datenrichtungsregister des 7501
1	$0001	Ein-/Ausgabe-Port des 7501
2	$0002	Flag für Schleifen
3- 4	$0003-0004	Neue Startadresse (Renumber)
5- 6	$0005-0006	Schrittweite (Renumber)
7	$0007	Gesuchtes Zeichen
8	$0008	Flag für Anführungszeichen-Modus
9	$0009	TAB-Spaltenzähler
10	$000A	Flag: 0 = Load, 1 = Verify

Tabelle 2. Die Zeropage beim C16

Dezimal	Hexadezimal	Bemerkung
11	$000B	Zeiger für Eingabepuffer, Anzahl der Elemente
12	$000C	Flag für Standard-DIM
13	$000D	Datentyp: $FF=String, $00=Numerisch
14	$000E	Datentyp: $80=Integer, $00=Fließkomma
15	$000F	Flag für DATA/LIST
16	$0010	Flag: Element/FNx-Flag
17	$0011	Flag: $00=INPUT, $40=GET, $98=READ
18	$0012	Flag: Vorzeichen des ATN
19	$0013	Flag: Input-Prompt
20- 21	$0014-0015	Int. Adresse
22	$0016	Zeiger auf temporären Stringstapel
23- 24	$0017-0018	Letzter temporärer Stringvektor
25- 33	$0019-0021	Stapel für temporäre Strings
34- 35	$0022-0023	Bereich für Hilfszeiger 1
36- 37	$0024-0025	Bereich für Hilfszeiger 2
38	$0026	Bereich für Produkt bei Multiplikation
39	$0027	Bereich für Produkt bei Multiplikation
40	$0028	Bereich für Produkt bei Multiplikation
41	$0029	Bereich für Produkt bei Multiplikation
42	$002A	Bereich für Produkt bei Multiplikation
43- 44	$002B-002C	Zeiger auf Basic-Anfang
45- 46	$002D-002E	Zeiger auf Variablen-Anfang
47- 48	$002F-0030	Zeiger auf Beginn der Arrays
49- 50	$0031-0032	Zeiger auf Ende der Arrays (+1)
51- 52	$0033-0034	Zeiger auf Stringspeicher
53- 54	$0035-0036	Hilfszeiger für Strings
55- 56	$0037-0038	Zeiger auf Speichergrenze
57- 58	$0039-003A	Laufende Basiczeilennummer
59- 60	$003B-003C	Textpointer
61- 62	$003D-003E	Zeiger auf Basic-Statement für CONT
63- 64	$003F-0040	Nummer der aktuellen DATA-Zeile
65- 66	$0041-0042	Adresse des aktuellen DATA-Elementes
67- 68	$0043-0044	Sprungvektor für INPUT
69- 70	$0045-0046	Aktueller Variablenname
71- 72	$0047-0048	Adresse der aktuellen Variablen
73- 74	$0049-004A	Variablenzeiger für FOR...NEXT
75- 76	$004B-004C	Zwischenspeicher für Basic-Zeiger
77	$004D	Akkumulator für Vergleichssymbole
78- 79	$004E-004F	Arbeitsbereich (Zeiger etc.)
80- 81	$0050-0051	Arbeitsbereich (Zeiger etc.)
82	$0052	Arbeitsbereich (Zeiger etc.)
83	$0053	Grafik-Modus
84- 86	$0054-0056	Sprungvektor für Funktionen
87- 96	$0057-0060	Bereich für numerische Operationen
97	$0061	Fließkomma-Akkumulator 1 (FAC): Exponent
98-101	$0062-0065	Fließkomma-Akkumulator 1 (FAC): Mantisse
102	$0066	Fließkomma-Akkumulator 1 (FAC): Vorzeichen
103	$0067	Zeiger für Polynom-Auswertung
104	$0068	Fließkomma-Akkumulator 1 Überlauf
105-110	$0069-006E	Fließkomma-Akkumulator 2 Exponent etc.
111	$006F	Vorzeichenvergleich Akku 1 mit Akku 2
112	$0070	Fließkomma-Akkumulator 1 niederwertige Stelle
113-114	$0071-0072	Kassettenpuffer Länge/Zeiger
115-116	$0073-0074	Automatisches Zeileninkrement 0=AUS
117	$0075	Grafik-Flag (0=Text, 255=Grafik)
118-120	$0076-0078	Arbeitsbereich
121-123	$0079-007B	Darstellung von DS$
124-125	$007C-007D	Basic-Pseudo-Stack-Pointer
126-143	$007E-008F	Arbeitsbereich (Sound)
144	$0090	Statuswort ST
145	$0091	Flag: Stop-Taste / RVS-Taste
148	$0094	Flag: Serieller Bus — Ausgabe Zeichenpuffer
149	$0095	Zeichenspeicher — Serieller Bus
150	$0096	Zwischenspeicher
151	$0097	Anzahl der geöffneten Files
152	$0098	Eingabegerät (Normalwert: 0)
153	$0099	Ausgabegerät (Normalwert: 3)
154	$009A	Flag: $80=Direkt-, $00=Programm-Modus
157-158	$009D-009E	Zeiger auf Ende des Programms
159-160	$009F-00A0	Temporärer Datenspeicher

Tabelle 2. Die Zeropage beim C16 (Fortsetzung)

Dezimal	Hexadezimal	Bemerkung
161-162	$00A1-00A2	Temporärer Datenspeicher
163-165	$00A3-00A5	Echtzeit-Uhr
166	$00A6	Register für seriellen Bus
167	$00A7	Register für Kassettenroutine
168	$00A8	Register für seriellen Bus
169	$00A9	Temporärer Farb-Vektor
170	$00AA	Register für Kassettenroutine
171	$00AB	Anzahl der Zeichen im Filenamen
172	$00AC	Aktuelle logische File-Nummer
173	$00AD	Aktuelle Sekundär-Adresse
174	$00AE	Aktuelle Geräte-Nummer
175-176	$00AF-00B0	Zeiger auf Filenamen
177	$00B1	Von Fehlerroutine benutzt
178-179	$00B2-00B3	I/O Startadresse
180-181	$00B4-00B5	Basis-Ladeadresse
182-183	$00B6-00B7	Zeiger: Anfang Kassettenpuffer
186-187	$00BA-00BB	Zeiger auf Zeichen im Kassettenpuffer
190-191	$00BE-00BF	Register für Long-Fetch-Routine
192-193	$00C0-00C1	Register für Scrolling
194	$00C2	RVS-Flag (Bildschirm)
195	$00C3	Zeiger: Ende der Zeile (Input)
196-197	$00C4-00C5	Cursorposition (Input), Reihe/Spalte
198	$00C6	Tastaturabfrage ($40=keine Taste)
199	$00C7	Flag: Eingabe Bildschirm/Tastatur
200-201	$00C8-00C9	Zeiger auf Bildschirmzeile
202	$00CA	Cursorposition in aktueller Bildschirmzeile
203	$00CB	Flag: Anführungszeichen-Modus ($00=nein)
204	$00CC	Länge der aktuellen Bildschirmzeile
205	$00CD	Zeile, in der sich der Cursor befindet
206	$00CE	Letztes Zeichen (I/O)
207	$00CF	Anzahl der Zeichen (Insert-Modus)
208-215	$00D0-00D7	Reserviert für Software
216-232	$00D8-00E8	Reserviert für Anwendungssoftware
233	$00E9	Arbeitsbereich
234-235	$00EA-00EB	Bildschirm-Editor (Farbe)
236-238	$00EC-00EE	Arbeitsbereich (Bildschirm)
239	$00EF	Anzahl der Zeichen im Tastaturpuffer
241-242	$00F1-00F2	Register für Monitor
250	$00FA	Register für X bei Stoptastentest
251	$00FB	Aktuelle Bank-Konfiguration
254	$00FE	Arbeitsregister (Editor)

Tabelle 2. Die Zeropage beim C16 (Schluß)

Dezimal	Hexadezimal	Bemerkung
0	$0000	Datenrichtungsregister des 8502
1	$0001	Datenregister des 8502
2	$0002	BANK-Teil des PC bei Registeranzeige (PC= Programmzähler)
3	$0003	MSB des PC bei Registeranzeige
4	$0004	LSB des PC
5	$0005	Prozessorstatus SR bei Registeranzeige
6	$0006	Akkumulator A bei Registeranzeige
7	$0007	X-Register X bei Registeranzeige
8	$0008	Y-Register Y bei Registeranzeige
9	$0009	Stapelzeiger SP bei Registeranzeige
9	$0009	Suchzeichen (sucht ':' oder Zeilenende)
10	$000A	Hochkommaflag
11	$000B	Spaltenspeicher beim TAB-Befehl
12	$000C	Flag: 0=Load, 1=Verify
13	$000D	Zeiger für Eingabepuffer, Anzahl der Elemente
14	$000E	Flag für Standard-DIN
15	$000F	Datentyp: $FF=String, $00=Numerisch
16	$0010	Datentyp: $80=Integer, $00=Fließkomma
17	$0011	Flag für DATA/LIST/Garbage-Collection
18	$0012	Flag: Element/FNxx-Flag
19	$0013	Flag: $00=INPUT,$40=GET,$98=READ
20	$0014	Flag: Vorzeichen des ATN
21	$0015	Aktuelles Ein-/Ausgabegerät
22- 23	$0016-$0017	Integerwert für Zeilennummer o. 2 Byte-Adr. für GOTO, GOSUB, POKE, PEEK, SYS, WAIT
24	$0018	Zeiger auf temporären Stringstapel

Tabelle 3. Die Zeropage beim C128

Dezimal	Hexadezimal	Bemerkung
25- 26	$0019-$001A	Letzter temporärer Stringvektor
27- 35	$001B-$0023	Stapel für temporäre Strings
36- 37	$0024-$0025	Bereich für Hilfszeiger 1
38- 39	$0026-$0027	Bereich für Hilfszeiger 2
40	$0028	Bereich für Produkt bei Multiplikation
41	$0029	Bereich für Produkt bei Multiplikation
42	$002A	Bereich für Produkt bei Multiplikation
43	$002B	Bereich für Produkt bei Multiplikation
44	$002C	Bereich für Produkt bei Multiplikation
45- 46	$002D-$002E	Zeiger auf Basic-Anfang
47- 48	$002F-$0030	Zeiger auf Variablen-Anfang
49- 50	$0031-$0032	Zeiger auf Beginn der Arrays
51- 52	$0033-$0034	Zeiger auf Ende der Arrays (+1)
53- 54	$0035-$0036	Zeiger auf Stringspeicher
55- 56	$0037-$0038	Hilfszeiger für Strings
57- 58	$0039-$003A	Zeiger auf Speichergrenze
59- 60	$003B-$003C	Laufend Basic-Zeilennummer
61- 62	$003D-$003E	Zeiger auf Basic-Text für CHRGET
63- 64	$003F-$0040	Hilfszeiger für PRINT USING
65- 66	$0041-$0042	Nummer der aktuellen DATA-Zeile
67- 68	$0043-$0044	Adresse des aktuellen DATA-Elementes
69- 70	$0045-$0046	Sprungvektor für INPUT
71- 72	$0047-$0048	Aktueller Variablenname
73- 74	$0049-$004A	Adresse der aktuellen Variablen
75- 76	$004B-$004C	Variablenzeiger für FOR...NEXT
77- 78	$004D-$004E	Zwischenspeicher für Basic-Zeiger
79	$004F	Akkumulator für Vergleichssymbole: $01=größer, $02=gleich, $04=kleiner
80- 81	$0050-$0051	Zeiger auf Variable einer DEF FN-Funktion
82- 84	$0052-$0054	Zeiger auf String-Descriptor in einer Variablen-Liste
85	$0055	Flag: HELP oder LIST
86- 88	$0056-$0058	Sprungvektor für Funktionen
89- 98	$0059-$0062	Bereich für numerische Operationen
99	$0063	Fließkomma-Akkumulator 1 (FAC): Exponent
100-103	$0064-$0067	Fließkomma-Akkumulator 1 (FAC): Mantisse
104	$0068	Fließkomma-Akkumulator 1 (FAC): Vorzeichen
105	$0069	Zeiger für Polynomauswertung
106-111	$006A-$006F	Fließkomma-Akkumulator 2 (Exponat,4 Byte Mantisse, Vorzeichenbyte)
112	$0070	Vorzeichenvergleich Akku 1 und Akku 2
113	$0071	Akku 1, niederwertige Stelle, Rundung
114-115	$0072-$0073	Kassettenpuffer Zeiger/Länge
116-117	$0074-$0075	Inkrement bei AUTO-Befehl, $00=aus
118	$0076	Grafik-Flag 0=Text, 255=Grafik
119-121	$0077-$0079	Arbeitsbereich
122-124	$007A-$007C	Darstellung von DS$
125-126	$007D-$007E	Basic-Pseudo-Stack-Pointer
127	$007F	Flag für Direktmodus oder Programm
128	$0080	Zeiger bei PRINT USING auf Dezimalpunkt
129	$0081	PARSTX
130	$0082	Enthält Stapelzeiger bei TRAP-Befehl
131	$0083	Aktuelle Zeichenfarbe
132	$0084	Aktuelle Multicolor-Farbe 1
133	$0085	Aktuelle Multicolor-Farbe 2
134	$0086	Aktuelle Vordergrundfarbe
135-136	$0087-$0088	X-Faktor bei SCALE-Befehl
137-138	$0089-$008A	Y-Faktor bei SCALE-Befehl
139	$008B	Flag für Randbegrenzung bei PAINT
140-141	$008C-$008D	Zeiger für Bitmap
142-143	$008E-$008F	Zwischenspeicher für Grafikoperationen
144	$0090	Statuswort ST
145	$0091	Flag: Stop-Taste
146	$0092	Zeit-Konstante für Kassette
147	$0093	Flag: $00=LOAD, $01=VERIFY
148	$0094	Flag: Serieller Bus - Ausgabe Zeichenpuffer
149	$0095	Zwischenpeicher - Serieller Bus
150	$0096	Zwischenpeicher (Kassetten SYNC.-Nr.)
151	$0097	Temporäre Datenadresse
152	$0098	Anzahl der offenen Dateien/Datentabellen-Index
153	$0099	Eingabegerät
154	$009A	Ausgabegerät
155	$009B	Paritätsbyte von Kassette
156	$009C	Flag: Byte empfangen
157	$009D	Flag: $80=Direkt-,$00=Programm-Modus
158	$009E	Bandfehler/Zeichenpuffer/Temporärer Speicher
159	$009F	Bandfehler korrigiert/Temporärer Speicher
160-162	$00A0-$00A2	Echtzeit-Uhr
163	$00A3	Bitzähler für serielle Ausgabe
164	$00A4	Zähler auf Band
165	$00A5	Startsynchronisation bei Kassette
166	$00A6	Zeiger im Kassettenpuffer
167-171	$00A7-$00AB	Flags für Schreiben/Lesen auf Kassette und RS232
172-173	$00AC-$00AD	Zeiger: Kassettenpuffer/Scrolling/Boot-Adresse
174-175	$00AE-$00AF	Zeiger auf Programmende bei LOAD
176-177	$00B0-$00B1	Bandzeitkonstante
178-179	$00B2-$00B3	Startadresse des Bandpuffers
180	$00B4	Bitzähler für Band und RS232
181	$00B5	Nächstes zu übertragendes Bit für Band und RS232
182	$00B6	RS232-Bytepuffer
183	$00B7	Länge des Filenamens
184	$00B8	Logische Filenummer
185	$00B9	Sekundäradresse
186	$00BA	Gerätenummer
187-188	$00BB-$00BC	Zeiger auf Filenamen
189	$00BD	Ein-/Ausgabespeicher (für seriell)
190	$00BE	Blockzähler für Band
191	$00BF	Puffer für serielle Ausgabe/Booten=$00
192	$00C0	Kassettenmotor-Flag
193-194	$00C1-$00C2	Startadresse für Ein-/Ausgabe beim Booten: Track(194) & Sektor (193)
195-196	$00C3-$00C4	Endadresse für Ein-/Ausgabe-Zwischenspeicher für Kernalvektoren
197	$00C5	Kassette: Schreib-/Lese-Daten
198	$00C6	Aktuelle Bank für LOAD/SAVE und VERIFY
199	$00C7	Bank, in der sich der Filename befindet
200-201	$00C8-$00C9	Zeiger auf RS232-Eingabepuffer
202-203	$00CA-$00CB	RS232-Ausgabe-Zeiger
204-205	$00CC-$00CD	Vektor: Tastatur-Decodiertabelle
206-207	$00CE-$00CF	Hilfszeiger für Strings
208	$00D0	Anzahl der Zeichen im Tastaturpuffer
209	$00D1	Länge eines Funktionstasten-Strings
210	$00D2	Zeiger auf String einer Funktionstaste
211	$00D3	Flag für SHIFT-/CTRL- und COMMODORE-Taste
212	$00D4	Nummer der augenblicklich gedrückten Taste
213	$00D5	Nummer der zuletzt gedrückten Taste
214	$00D6	RETURN-Input-Flag
215	$00D7	Flag für Bildschirmformat ($00=40Zeichen, $80=80Zeichen)
216	$00D8	Flag für Grafikmodus
217	$00D9	RAM/ROM-Flag für Video-Chip
218-219	$00DA-$00DB	Hilfszeiger für Bildschirmeditor (Scrolling etc.)
220-221	$00DC-$00DD	Hilfszeiger für Bildschirmeditor (Scrolling etc.)
222	$00DE	Temporäre Speicher für Editor
223	$00DF	Temporäre Speicher für Editor
224-225	$00E0-$00E1	Zeiger der aktuellen Zeile im Bildschirm-RAM
226-227	$00E2-$00E3	Zeiger der aktuellen Zeile im Attribut-RAM
228	$00E4	Fenster: Unterer Rand
229	$00E5	Fenster: Oberer Rand
230	$00E6	Fenster: Linker Rand
231	$00E7	Fenster: Rechter Rand
232	$00E8	Startspalte bei Eingabe
233	$00E9	Startzeile bei Eingabe
234	$00EA	Schlußzeile bei Eingabe
235	$00EB	Aktuelle Cursor-Zeile
236	$00EC	Aktuelle Cursor-Spalte
237	$00ED	Maximale Anzahl der Zeilen (24)
238	$00EE	Maximale Anzahl der Spalten (39 oder 79)
239	$00EF	Nächstes auszugebendes Zeichen
240	$00F0	Vorhergehendes Zeichen (für ESC-Test)
241	$00F1	Attribut des nächsten Zeichens (Farbe)
242	$00F2	Zwischenspeicher für Farbe
243	$00F3	Reverse-Flag
244	$00F4	Flag für Anführungszeichen

Tabelle 3. Die Zeropage beim C 128 (Fortsetzung)

Dezimal	Hexadezimal	Bemerkung
245	$00F5	Flag für Insert-Modus
246	$00F6	Flag für automatisches Einfügen
247	$00F7	Sperrt COMMODORE/SHIFT und CTRL-S
248	$00F8	Verhindert Bildschirmscrollen und Zeilenverknüpfung
249	$00F9	Verhindert CTRL-G (Bell)
250-255	$00FA-$00FF	freie Speicherstellen

Tabelle 3. Die Zeropage beim C128 (Schluß)

Dezimal	Hexadezimal	Bemerkung
256- 511	$0100-01FF	Prozessorstack
512- 600	$0200-0258	Eingabepuffer
601- 604	$0259-025C	Basic-Puffer
605- 684	$025D-02AC	Basic-/DOS-Arbeitsbereich
605	$025D	DOS-Schleifenzähler
606- 621	$025E-026D	Bereich für Filenamen
622	$026E	1. Filename (Länge)
623	$026F	DOS (Laufwerk 1)
624- 625	$0270-0271	1. Filename (Adresse)
626	$0272	2. Filename (Länge)
627	$0273	DOS (Laufwerk 2)
628- 629	$0274-0275	2. Filename (Adresse)
630	$0276	DOS logische Adresse
631	$0277	DOS (Geräteadresse)
632	$0278	DOS (Sekundäradresse)
633- 634	$0279-027A	DOS (Disketten-ID)
635	$027B	ID-Flag
636	$027C	DOS (Ausgabepuffer)
637- 684	$027D-02AC	DOS (Arbeitsbereich)
685- 688	$02AD-02B0	Grafik-Cursor
689- 692	$02B1-02B4	Grafik-Cursor (Register)
693- 715	$02B5-02CB	Grafik-Register
716- 739	$02CC-02E3	Print-Using, Grafik-Arbeitsbereich
740	$02E4	High-Byte-Adresse des Charakter-ROM
747	$02EB	Trace-Flag
752	$02F0	Anzahl der Grafik-Parameter
753	$02F1	Parameter: Relativ (1), Absolut (0)
754- 755	$02F2-02F3	Fließkomma-Vektor
756- 757	$02F4-02F5	Integer-Vektor
768- 769	$0300-0301	Sprungvektor: Fehlermeldung
770- 771	$0302-0303	Sprungvektor: Basic-Warmstart
772- 773	$0304-0305	Sprungvektor: Token-Generierung
774- 775	$0306-0307	Sprungvektor: Keywort erzeugen
776- 777	$0308-0309	Sprungvektor: Hauptschleife
778- 779	$030A-030B	Sprungvektor: Eval
780- 781	$030C-030D	Sprungvektor: Token-Generierung (User)
782- 783	$030E-030F	Sprungvektor: Keywort erzeugen
784- 785	$0310-0311	Sprungvektor: User-Token bearbeiten
786- 787	$0312-0313	Sprungvektor: Interrupt (Uhr)
788- 789	$0314-0315	Sprungvektor: Interrupt
790- 791	$0316-0317	Sprungvektor: Break Interrupt
792- 793	$0318-0319	Sprungvektor: Open
794- 795	$031A-031B	Sprungvektor: Close
796- 797	$031C-031D	Sprungvektor: Kanal für Eingabe öffnen
798- 799	$031E-031F	Sprungvektor: Kanal für Ausgabe öffnen
800- 801	$0320-0321	Sprungvektor: I/O zurücksetzen
802- 803	$0322-0323	Sprungvektor: Input
804- 805	$0324-0325	Sprungvektor: Ausgabe
806- 807	$0326-0327	Sprungvektor: Abfrage der Stoptaste
808- 809	$0328-0329	Sprungvektor: Getin-Routine
810- 811	$032A-032B	Sprungvektor: Schließen aller Files
812- 813	$032C-032D	Sprungvektor: Monitor Break
814- 815	$032E-032F	Sprungvektor: Lade-Routine
816- 817	$0330-0331	Sprungvektor: Save-Routine
818-1010	$0332-03F2	Kassettenpuffer
1011-1012	$03F3-03F4	Datenzähler (Write)
1013-1014	$03F5-03F6	Datenzähler (Read)
1015-1078	$03F7-0436	RS232-Input-Puffer (64 Byte)
1079-1108	$0437-0454	System-Arbeitsbereich

Tabelle 4. Die restlichen Systemadressen beim C16

Dezimal	Hexadezimal	Bemerkung
1109-1138	$0455-0472	System-Arbeitsbereich
1139-1144	$0473-0478	Chrget-Routine
1145-1156	$0455-0484	Chrgot-Routine
1172-1244	$0494-04DC	Bankswitching-Routinen
1255-1258	$04E7-04EA	Print-Using-Parameter
1263-1270	$04EF-04F6	Trap- und Error-Flags
1280-1282	$0500-0502	USR-Sprungbefehl
1283-1287	$0503-0507	Startwert für RND
1288	$0508	Flag für Kalt- oder Warmstart
1289-1298	$0509-0512	Tabelle der logischen Filenummern
1299-1308	$0513-051C	Tabelle der Gerätenummern
1309-1318	$051D-0526	Tabelle der Sekundäradressen
1319-1328	$0527-0530	Tastaturpuffer
1329-1330	$0531-0532	Basic-Anfang
1331-1332	$0533-0534	Basic-Ende
1337	$0539	Zeiger: Kassettenpuffer
1338	$053A	Typ des Kassettenfiles
1344	$0540	Tasten-Wiederholfunktion ($80=alle, $40=keine, $00=nur DEL, CUR, SPC)
1345-1346	$0541-0542	Zähler für Wiederholfunktion
1347	$0543	Shift-Flag
1348	$0544	Letztes Shift-Zeichen
1349-1350	$0545-0546	Sprungvektor: Keylog
1351	$0547	Text-/Grafik-Flag
1352	$0548	Scroll-Flag
1355-1372	$054B-0551	Cpu-Arbeitsbereich
1362-1367	$0552-0557	Cpu-Register
1368-1372	$0558-055C	Cpu-Arbeitsbereich
1373	$055D	Zähler für Funktionstasten
1374	$055E	Zeiger: Text, Funktionstasten
1375-1510	$055F-05E6	Speicher für Funktionstasten
1516-1571	$05EC-06EB	1 Page, Bankingroutinen
1516-1519	$05EC-05EF	Adreßtabelle
1520-1521	$05F0-05F1	Long-Jump (Adresse)
1522	$05F2	Long-Jump (Akkumulator)
1523	$05F3	Long-Jump (X-Register)
1524	$05F4	Long-Jump (Status-Register)
1525-1629	$05F5-065D	RAM-Bereich für Bankswitching
1630-1771	$065E-06EB	RAM-Bereich für Benutzersoftware
1792-1967	$06EC-07AF	Basic-Pseudo-Stack
1968-1996	$07B0-07CC	Kassetten-Arbeitsbereich
1997-2000	$07CD-07D0	RS232-Arbeitsbereich
2001	$07D1	RS232: Pointer auf Anf. Eingabepuffer
2002	$07D2	RS232: Pointer auf Ende Eingabepuffer
2003	$07D3	Anzahl der Zeichen im Eingabepuffer
2021	$07E5	Bildschirm: Unterer Rand
2022	$07E6	Bildschirm: Oberer Rand
2023	$07E7	Bildschirm: Linker Rand
2024	$07E8	Bildschirm: Rechter Rand
2026	$07EA	$FF=Automatisches Einfügen
2030-2033	$07EE-07F1	Bit-Tabelle
2034	$07F2	A-Register retten bei Sys-Kommando
2035	$07F3	X-Register retten bei Sys-Kommando
2036	$07F4	Y-Register retten bei Sys-Kommando
2037	$07F5	Status-Register retten bei Sys-Kommando
2038	$07F6	Register für Tastaturabfrage
2039	$07F7	Flag: CTRL-S: $00=offen, $06=gesperrt
2040	$07F8	RAM-ROM-Umschaltung für Monitor 0=ROM, $80=RAM
2044	$07FC	Motorsteuerung

Tabelle 4. Die restlichen Systemadressen beim C16 (Schluß)

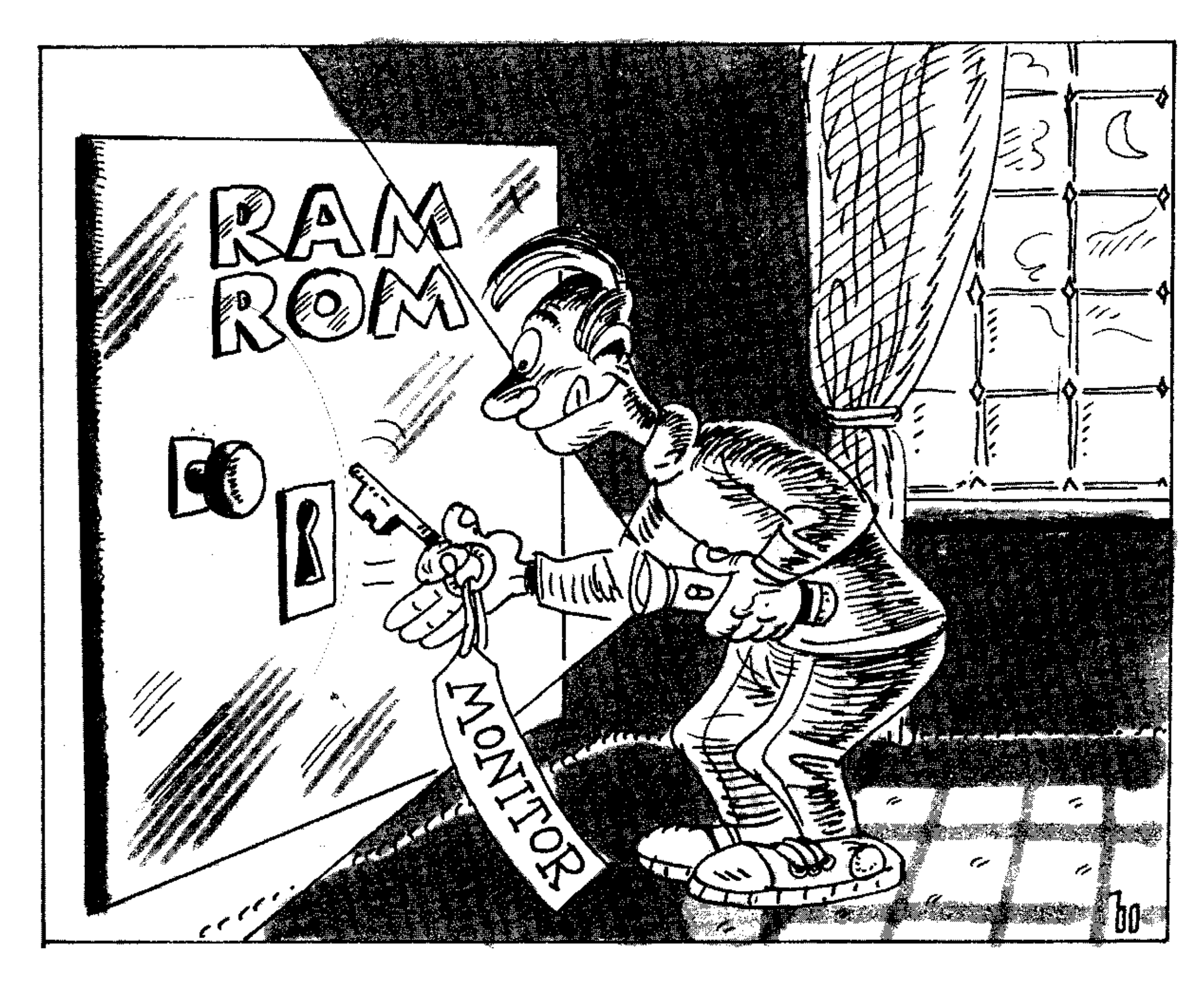
Dezimal	Hexadezimal	Bemerkung
256-271	$0100-$010F	Bereich für Filename
272	$0110	Zähler für DOS
273	$0111	Länge des 1. Filenamens
274	$0112	DOS (Laufwerk 1)
275	$0113	Länge des 2. Filenamens
276	$0114	DOS (Laufwerk 2)

Tabelle 5. Die restlichen Systemadressen beim C128

Dezimal	Hexadezimal	Bemerkung
277-278	$0115-$0116	Adresse des 2. Filenamens
279-280	$0117-$0118	Startadresse BLOAD/BSAVE
281-282	$0119-$011A	Endadresse BSAVE
283	$011B	DOS logische Adresse
284	$011C	DOS Geräteadresse
285	$011D	DOS Sekundäradresse
286	$011E	DOS Record-Länge
287	$011F	DOSBNK
288-289	$0120-$0121	DOS Disketten-ID
290	$0122	Flag: Arbeit beendet
291-310	$0123-$0136	Adressen für PRINT USING
291	$0123	Zeiger auf Beginn
292	$0124	Zeiger auf Ende
293	$0125	Flag: Dollar-Zeichen
294	$0126	Flag: Komma
295	$0127	Zähler
296	$0128	Vorzeichen Exponent
297	$0129	Zeiger auf Exponent
298	$012A	Anzahl der Zahlen vor dem Punkt
299	$012B	Justierungs-Flag
300	$012C	Anzahl der Positionen vor dem Punkt
301	$012D	Anzahl der Positionen nach dem Punkt
302	$012E	Flag: +/-
303	$012F	Flag: Exponent
304	$0130	Schalter
305	$0131	Zeichenzähler
306	$0132	Vorzeichennummer
307	$0133	Flag: */SPACE
308	$0134	Zeiger: Anfang des Feldes
309	$0135	Länge des Formatstrings
310	$0136	Zeiger: Ende des Feldes
311-511	$0137-$01FF	System-Stack
512-763	$0200-$02FB	Basic-Puffer/Monitor-Puffer
764-765	$02FC-$02FD	Sprungvektor: zusätzliche Funktionsroutinen
766-767	$02FE-$02FF	Sprungvektor: Function-Cartridge
768-769	$0300-$0301	Sprungvektor: Fehlermeldung
770-771	$0302-$0303	Sprungvektor: Basic-Warmstart
772-773	$0304-$0305	Sprungvektor: Token-Generierung
774-775	$0306-$0307	Sprungvektor: Keywort erzeugen
776-777	$0308-$0309	Sprungvektor: Befehl ausführen
778-779	$030A-$030B	Sprungvektor: Token auswerten
780-781	$030C-$030D	Sprungvektor: Escape-Umwandlung
782-783	$030E-$030F	Sprungvektor: Escape List
784-785	$0310-$0311	Sprungvektor: Escape ausführen
786-787	$0312-$0313	Sprungvektor: Interrupt (Uhr)
788-789	$0314-$0315	Sprungvektor: Interrupt (IRQ)
790-791	$0316-$0317	Sprungvektor: Interrupt (BRK)
792-793	$0318-$0319	Sprungvektor: Interrupt (NMI)
794-795	$031A-$031B	Sprungvektor: OPEN
796-797	$031C-$031D	Sprungvektor: CLOSE
798-799	$031E-$031F	Sprungvektor: Kanal für Eingabe öffnen
800-801	$0320-$0321	Sprungvektor: Kanal für Ausgabe öffnen
802-803	$0322-$0323	Sprungvektor: I/O zurücksetzen
804-805	$0324-$0325	Sprungvektor: INPUT
806-807	$0326-$0327	Sprungvektor: Ausgabe
808-809	$0328-$0329	Sprungvektor: Abfrage der Stop-Taste
810-811	$032A-$032B	Sprungvektor: GETIN-Routine
812-813	$032C-$032D	Sprungvektor: Schließen aller Files
814-815	$032E-$032F	Sprungvektor: Monitor BREAK
816-817	$0330-$0331	Sprungvektor: LOAD-Routine
818-819	$0332-$0333	Sprungvektor: SAVE-Routine
927-977	$039F-$03D1	Unterprogramme zum Laden aus beliebiger Bank
978-980	$03D2-$03D4	Konstante für Basic
981	$03D5	Bank für PEEK, POKE, SYS...
982-985	$03D6-$03D9	Temporäre Register
986	$03DA	Bank-Zeiger für Zahl/String-Umwandlung
987-990	$03DB-$03DE	Temporäre Speicher für SSHAPE
991	$03DF	BITS
992-993	$03E0-$03E1	Temporärer Speicher für SPRSAV
994	$03E2	Vordergrund-/Hintergrund-Farbe
995	$03E3	Vordergrund-/Multicolor1-Farbe
1024-2047	$0400-$07FF	Bildschirmspeicher im 40-Zeichen-Modus
2048-2559	$0800-$09FF	Basic-Run-Time-Stack
2560-2561	$0A00-$0A01	Sprungvektor: Basic-Warmstart
2562-2563	$0A02-$0A03	Sprungvektor: Kaltstart
2563	$0A03	Flag: PAL/NTSC
2564	$0A04	Flag: RESTE/NMI
2565-2566	$0A05-$0A06	Zeiger: Anfang verfügbarer Speicher in System-Bank
2567-2568	$0A07-$0A08	Zeiger: Ende verfügbarer Speicher in System-Bank
2569-2570	$0A09-$0A0A	Sprungvektor: IRQ (Kassette)
2571	$0A0B	Zeitvergleich bei Kassettenoperationen
2572-2573	$0A0C-$0A0D	Temporäre Speicher beim Kassetten-Lesen
2574	$0A0E	Flag: Timeout
2575-2587	$0A0F-$0A1B	Arbeitsbereich RS232
2588	$0A1C	Flag: Intern/Extern
2589-2591	$0A1D-$0A1F	Zwischenspeicher Echtzeituhr
2592	$0A20	Zeiger: Tastatur-Warteschlange
2593	$0A21	Flag: CTRL-5
2594	$0A22	Flag: REPEAT $80=alle Tasten, $40=keine Tasten, $00=Cursortasten/DEL/INST
2595-2596	$0A23-$0A24	Zähler: REPEAT
2597	$0A25	Zähler: COMMODORE/SHIFT
2598	$0A26	Flag für Cursor-Modus, $40=fester Cursor
2599-2600	$0A27-$0A28	Hilfsregister für VIC
2601	$0A29	Cursorzeichen vor dem Blinken
2602	$0A2A	Cursorfarbe vor dem Blinken
2603	$0A2B	Cursormodus 80 Zeichen (VDC):$80=fest,$60=normal blinkend, $40=schnell blinkend, $20=Aus
2604-2607	$0A2C-$02AF	Basiszeiger VIC/VDC
2608	$0A30	Temporäre Zeiger auf letzte Zeile
2609-2612	$0A31-$0A34	Temporäre Register (VDC)
2613	$0A35	Cursorfarbe vor dem Blinken (VDC)
2614	$0A36	Splitscreen-Raster-Wert (VIC)
2615	$0A37	Zwischenspeicher
2616	$0A38	Hilfszähler für Echtzeituhr (PAL)
2688-2739	$0A80-$0AB4	Systemadressen für Monitor
2752	$0AC0	Aktuelle Funktionstasten ROM-Bank
2753	$0AC1	Geräteadresse
2816-3071	$0B00-$0BFF	Kassettenpuffer und Puffer für BOOT-Sektor
3072-3327	$0C00-$0CFF	RS232-Eingabepuffer
3328-3583	$0D00-$0DFF	RS232-Ausgabepuffer
3584-4095	$0E00-$0FFF	Bereich für Sprite-Definition
4096-4351	$1000-$10FF	Bereich für programmierbare Funktionstasten
4352-4400	$1100-$1130	Bereich für DOS-String
4401-4408	$1131-$1138	Koordinaten für Grafikcursor
4409-4424	$1139-$1149	Parameter für LINE
4425-4431	$1149-$114E	Variablen für Winkel-Routinen
4432-4439	$1150-$1157	Variablen für CIRCLE-Befehl
4432-4447	$1150-$115F	Variablen für BOX-Befehl
4433-4449	$1151-$1162	Variablen für SSHAPE/GSHAPE
4456-4566	$1168-$11D6	Allgemeine Grafikvariablen
4608-4609	$1200-$1201	Vorhergehende Basic-Zeilennummer
4610-4611	$1202-$1203	Zeiger auf Basic-Befehl (für CONT)
4612-4615	$1204-$1207	Symbole bei PRINT USING
4616-4621	$1208-$120D	Parameter für Fehlerbehandlung
4622-4641	$12DE-$1221	Allgemeine Basic-Variablen
4642-4725	$1222-$1275	Variablen für Musikbefehle
4726-4736	$1276-$1280	diverse Interrupt-Flags
4737-4784	$1281-$12B0	Variablen für SOUND-Befehle
4785-4860	$12B1-$12FC	Register für Paddles, WINDOW und SPRDEF
4864-6143	$1300-$17FF	Unbenutzter RAM-Bereich
7168-8191	$1C00-$1FFF	Start Basic-Text/bei Grafik: Video-Matrix
8192-16383	$2000-$3FFF	VIC-Bitmap (im Grafikmodus)
16384	$4000	Start Basic-Text (im Grafikmodus)
53248	$D000	Zeichensatz/VIC-Register
54272	$D400	SID-Register
54528	$D500	Register der MMU im I/O-Bereich
54784	$D600	80-Zeichen-Videochip
55296	$D800	VIC Color-Nibble
56320	$DC00	CIA 1
56576	$DD00	CIA 2
56832	$DE00	Expansion-Port I/O
57088	$DF00	Expansion-Port I/O (für DMA)
65280	$FF00	Register der MMU

Tabelle 5. Die restlichen Systemadressen beim C128 (Schluß)

Durchblick mit dem Monitor



Vielen C16/116-Anwendern ist der eingebaute Maschinensprache-Monitor TEDMON immer noch ein Buch mit sieben Siegeln. Hier soll Ihnen gezeigt werden, was man mit ihm als Basic-Programmierer alles anfangen kann.

Leider wird der TEDMON im C16-Handbuch nur für Maschinensprache-Freaks erklärt. Basic-Programmierer werden praktisch auf dem Trockenen sitzen gelassen. Das aber eigentlich zu Unrecht, denn auch der Basic-Programmierer kann mit diesem Werkzeug eine ganze Menge anfangen – auch ohne Kenntnisse in Maschinensprache.

Was ist überhaupt ein Monitor? Diese Frage hat zwei Antworten. Die erste kennen Sie bestimmt: Ein Sichtgerät, an das Sie Ihren Computer anschließen können. Die zweite Antwort ist: Keine Hardware, sondern ein Programm, mit dem man einen Einblick in den Speicher des Computers bekommt.

Im C16, C116 und Plus/4 ist ein solches Monitorprogramm schon integriert: der TEDMON. Bevor wir mit diesem arbeiten können, müssen wir uns noch ein wenig Grundlagenwissen aneignen.

Die grundlegende Fähigkeit eines Computers ist es, Daten erfassen zu können. Man spricht beim Menschen davon, daß er sich etwas merkt. Beim Computer sagt man, er speichert etwas. Daher kommt der Begriff Speicher.

Der Speicher benötigt eine bestimmte Struktur, um zweckmäßig arbeiten zu können. Wir schlüsseln diese Unterteilung in Speichereinheiten »von unten« auf, beginnen also mit der kleinsten Speichermenge. Die einfachste Art von Information ist die Unterscheidung zwischen zwei Möglichkeiten (ja/nein, gut/schlecht, und so weiter).

Beim Computer, der sich Speicherwerte (vereinfacht gesagt) durch Verändern der elektrischen Spannung merkt, wird zwischen »Strom fließt« und »Strom fließt nicht« unterschieden. Dabei steht »0« für »kein Strom« und »1« für »Strom«.

Diese primitivste Form der Information nennt man sowohl in der Informatik (»Computerlehre«) als auch in der modernen Informationstheorie »Bit«. »BIT« kommt vom englischen »BInary digiT« (Binärziffer), da in einem Zahlensystem der Basis 2 nur 0 und 1 als Ziffern vorhanden sind (zum Vergleich: Wir arbeiten im alltäglichen Gebrauch mit der Basis 10, dem Dezimalsystem).

Natürlich ist ein Bit zu wenig, um vernünftig arbeiten zu können. Deshalb kombiniert man mehrere Bits, das heißt man nimmt mehrere Bits zu einer höheren Informationseinheit zusammen und betrachtet jedes einzelne Bit als Stelle einer Binärzahl. Mathematisch gesehen haben wir dann »2 hoch Anzahl der Bit«-Kombinationen.

Computerhersteller und -entwickler kamen zu der Lösung, 8 Bit als ein »Byte« zu bezeichnen. Wir können also 2^8, das sind 256 verschiedene Werte darstellen. Hier ein paar Beispiele:

binär %01010101 = dezimal 85
binär %11111111 = dezimal 255 (höchste Zahl, die mit 8 Bit dargestellt werden kann)
binär %00000000 = dezimal 0 (niedrigste mit 8 Bit darstellbare Zahl)
binär %10001000 = dezimal 136

Ein Byte kann alle Zahlen von 0 bis 255, also 256 verschiedene Werte, annehmen. Eine »Speicherzelle« beim Computer hat genau 1 Byte Kapazität. Bei den Beispielen der Binärzahlen ist Ihnen vielleicht das Prozentzeichen »%« vor jedem Binärwert aufgefallen. Dieses Zeichen setzt man in der Regel vor eine Binärzahl, um damit das Zahlensystem der Basis 2 zu kennzeichnen

Das Zusammenfassen von Bits in einem Byte erleichtert uns die Einteilung des Speichers erheblich; man darf aber nicht außer acht lassen, daß für den Computer nach wie vor der gesamte Inhalt des Speichers nur eine Ansammlung sehr vieler Bits ist.

Auch Buchstaben und andere Zeichen werden vom Computer als Byte behandelt. Dafür gibt es eine Codierung, nach der jedem Byte-Wert ein Zeichen zugewiesen ist, zum Beispiel steht die Zahl 65 (dezimal) für den Buchstaben A, 66 für B, 67 für C, etc.. Diese Codierung trägt die Bezeichnung »ASCII« (»American Standard Code for Information Interchange« oder deutsch »Amerikanische Standard-Codierung für den Informationsaustausch«). Man sagt also zum Beispiel: »Der ASCII-Code von A ist 65«. Soviel zunächst zum ASCII-Code; dazu können Sie sich auch im Handbuch zu Ihrem Computer informieren.

Natürlich ist ein einziges Byte zu wenig. Von einem Computer erwartet man, daß er viele Zeichen aufnehmen kann. Unser Computer hat zum Beispiel über 16000 Byte Speicherkapazität, wenn es sich um einen C16 oder C116 handelt. Der Plus/4 verfügt sogar über ganze 65536 Byte Speicher.

Damit der Computer »weiß«, welche Speicherstelle angewählt (zum Lesen oder zum Schreiben) werden soll, erhält jede Speicherzelle eine ihr zugewiesene Nummer, die sogenannte »Speicheradresse« oder einfach nur »Adresse«. Mit der Nummer 1024 wird also beispielsweise die 1025. Speicherstelle im Computer angesprochen. 1025 deshalb, weil die Zählung im Computer immer mit Null beginnt. Die erste Speicherstelle trägt also die Adresse 0.

Eine Einheit von $2^{10} = 1024$ Byte nennt man »Kilo-Byte« oder kurz »KByte«. Die Mengeneinheit »Kilo« steht also nicht für 1000, sondern für 1024 Byte. Das ist übrigens eine häufige Fehlerquelle. Wenn man von einem Computer mit 64 KByte Speicher spricht, dann handelt es sich nicht um 64000 Byte, sondern um 64 * 1024 = 65536 Byte Speicherplatz. Da der Computer intern binär arbeitet, also keine 1000 als gerade Potenz (im Dezimalsystem 10^3) kennt, ist diese Einteilung zweckmäßig.

Da unser C16 in der Grundausführung 16 KByte sogenanntes »RAM« zur Verfügung hat, sind das also 16 * 1024 = 16384 Byte.

Was ist »RAM«?

Wir sprachen eben von »RAM«. Das bedarf einer kurzen Erklärung. RAM (vom englischen »Random Access Memory« oder auf deutsch etwa »Speicher mit wahlfreiem Zugriff«) ist der Speicher, dessen Inhalt man im Computer verändern kann. Ein Basic-Programm oder Ihre Daten werden beispielsweise im RAM abgelegt beziehungsweise gespeichert.

Die andere Sorte von Speicher ist das ROM (»Read Only Memory«), der Nur-Lese-Speicher. Im ROM stehen bestimmte Daten fest eingespeichert (die elementaren Steuerprogramme des Computers, die so etwas wie Bildschirmausgabe oder Basic-Programmierung erst ermöglichen) und können zwar gelesen, aber nicht geändert werden. Der Vorteil des ROM gegenüber des RAM ist, daß die dort enthaltenen Werte auch nach dem Ausschalten der Stromzufuhr erhalten bleiben. Der Vorteil des RAM ist, daß man freien Zugriff (sowohl Lesen als auch Schreiben von Speicherinhalten) auf den Speicher hat.

Im Computer werden dabei in der Regel beide Möglichkeiten realisiert. Das RAM steht dem Anwender für seine Arbeit zur Verfügung (bis auf wenige Ausnahmen), und das ROM enthält das Steuerprogramm, das den Computer nach dem Einschalten in einen kontrollierten Zustand bringt. Dieses Steuerprogramm nennt man auch Betriebssystem, wobei dieses Betriebssystem für die elementaren Vorgänge, wie Schreiben auf den Bildschirm, Eingabe von der Tastatur, Laden und Speichern von Programmen, zuständig ist.

Das Hexadezimalsystem

Wir wissen nun, daß der Computer intern binär rechnet, auch wenn er in Basic uns gegenüber so tut, als ob er dezimal arbeitet. Für diesen Komfort sind extra Rechenprogramme verantwortlich, die im ROM des Computers gespeichert sind.

Der Monitor führt uns aber näher an die Maschinenebene heran, so daß wir uns dem Computer ein wenig anpassen müssen. Es wäre jedoch zuviel verlangt, wenn wir jetzt auf einmal in Nullen und Einsen denken sollten (vor wenigen Jahren war das jedoch noch so!). Andererseits müssen wir aber auf den Komfort unter Basic verzichten.

Der Kompromiß ist ein Zahlensystem, das leicht vom Computer ins Binärsystem umgewandelt, aber auch von uns leicht verwendet werden kann. So »erfand« man das Hexadezimalsystem, ein Zahlensystem mit der Basis 16. Die Vereinfachung ist dabei, daß je 4 Bit einer hexadezimalen Ziffer entsprechen. 1 Byte (= 8 Bit) kann also aus maximal 2 Hex-Ziffern bestehen ($2^8 = 16^2$).

Es stehen die vom Dezimalsystem her bekannten Ziffern 0 bis 9 weiterhin zur Verfügung; außerdem brauchen wir noch Ziffern, die den dezimalen Werten 10 bis 15 entsprechen. Hierfür stehen die Buchstaben A bis F. Es ergibt sich für Hexadezimal-Ziffern folgende Umrechnungstabelle:

Hexadezimal	Dezimal	Hexadezimal	Dezimal
0	0	8	8
1	1	9	9
2	2	A	10
3	3	B	11
4	4	C	12
5	5	D	13
6	6	E	14
7	7	F	15

Treffen wir an dieser Stelle eine Vereinbarung: Um Dezimalzahlen von Hexadezimal-Zahlen (auch: Hex-Zahlen) zu unterscheiden, stellen wir den Hex-Zahlen immer ein Dollarzeichen »$« voran. $44 ist also hexadezimal, 44 (ohne Dollar davor) dezimal. Dieses System kennen Sie aber schon von den Binärzahlen her, als wir ein Prozentzeichen »%« zur Unterscheidung verwendeten.

Selbstredend können Hex-Zahlen auch aus mehreren Stellen bestehen.

Im Dezimalsystem ist:

$345 = 5 * 10^0 + 4 * 10^1 + 3 * 10^2$

Zur Probe:

$5 * 10^0 + 4 * 10^1 + 3 * 10^2$

$= 5 * 1 + 4 * 10 + 3 * 100$

$= 5 + 40 + 300$

$= 345$

Ähnlich gehen wir im Hexadezimalsystem vor, allerdings ist die Basis jetzt 16 und wir müssen die Hex-Ziffern umrechnen:

$\$3A4 = \$4 * \$10^0 + \$A * \$10^1 + \$3 * \$10^2$

$= 4 * 16^0 + 10 * 16^1 + 3 * 16^2$

$= 4 * 1 + 10 * 16 + 3 * 256$

$= 4 + 160 + 768$

$= 932$

Beim Arbeiten mit dem Monitor wird alles über das hexadezimale Zahlensystem abgewickelt: Der Monitor gibt Werte (Byte, Adressen) hexadezimal aus, wir geben Werte hexadezimal ein.

Sollte einmal eine Umrechnung anfallen, so greifen wir auf Basic zurück. Den hexadezimalen Wert einer Dezimalzahl erhalten wir über

```
PRINT HEX$(Dezimalzahl)
```

zum Beispiel

```
PRINT HEX$(952)
```

Die gegensätzliche Umrechnung erfolgt über

```
PRINT DEC("Hex-Zahl")
```

zum Beispiel

```
PRINT DEC("5FA3")
```

Dabei sind Werte von 0-65535 beziehungsweise $0000-$FFFF zugelassen, andere Werte benötigen wir auch nicht.

Der Monitor kann auf drei Arten gestartet werden, von denen nur zwei für uns interessant sind:

1) Über den Maschinensprachebefehl »BRK«. Diese Variante vernachlässigen wir, denn uns interessiert hier nur Basic.

2) RESET und gleichzeitig RUN/STOP-Taste gedrückt halten. Dies bewirkt, daß beim RESET nicht ins Basic, sondern in den Monitor gesprungen wird.

3) Eingabe des Basic-Befehls <MONITOR>.

Mit welcher der drei genannten Möglichkeiten der Monitor auch gestartet wird, er meldet sich mit einer Einschaltmeldung:

```
MONITOR
  PC   SR AC XR YR SP
; 0000 00 00 00 00 F8
```

»MONITOR« signalisiert, daß wir uns nun im Monitor befinden. Jetzt haben alle Basic-Kommandos ihre Gültigkeit verloren, bis wir den Monitor verlassen. Dafür gelten nun die Monitor-Anweisungen.

Start des Monitors

Die weiteren Zeilen der Einschaltmeldung können wir vernachlässigen, denn die sind nur für Maschinensprache-Anwendungen interessant. In diesem Kurs werden wir aber nichts besprechen, was nur für Maschinensprache Bedeutung hat; wer Maschinensprache kann, findet im 64'er-Sonderheft 3/1986 ab Seite 14 und auch in diesem Sonderheft eine ganze Menge Informationen darüber.

Der Eingabemodus des Monitors ist nicht anders als in Basic, Cursortasten und so weiter funktionieren weiterhin.

Nur die Basic-Anweisungen und Zeilennummern gibt es jetzt nicht mehr.

Was versteht er aber dann für eine »Sprache«?

Dazu wollen wir gleich unseren ersten Befehl lernen. Er dient zum Verlassen des Monitors (ins Basic) und heißt ganz einfach »X«. Sie wissen schon, wie man dieses Kommando ausführt: <X> eingeben und <RETURN> drücken. Das vertraute »READY.« erscheint und wir sehen daran, daß wir uns im Basic befinden. Nun können wir wieder Basic-Befehle ausführen lassen. Ein Neustart des Monitors erfolgt über <MONITOR>.

Dem Befehl »X« müssen keine weiteren Angaben (sogenannte Parameter) folgen, der Monitor weiß auch so, was er zu tun hat.

In Bild 1 finden Sie eine Kurzübersicht zum X-Befehl. Wir werden Ihnen zu jedem neuen Befehl eine solche Kurzübersicht liefern, damit Sie bei Bedarf schnell nachschlagen können.

Übrigens bestehen alle Monitor-Kommandos – wie der X-Befehl – nur aus einem einzigen Zeichen. Dieses Zeichen ist meist aus einem englischen Wort, das in der Kurzübersicht aufgeführt ist, abgeleitet.

Der nächste Befehl, den wir kennenlernen, ist nicht so einfach wie »X« anzuwenden. Dafür eröffnet uns dieses Kommando aber auch wesentlich mehr Möglichkeiten. Es ist der M-Befehl, der »Grundbefehl« eines Monitors. Wenn wir uns mit diesem vertraut gemacht haben, sind die restlichen Befehle schnell erlernt. Lassen Sie sich beim Verstehen des M-Befehls also Zeit, es lohnt sich!

Zunächst etwas Grundsätzliches: Mit <RUN/STOP> können fast alle Monitor-Befehle abgebrochen werden. Bei »X« ist dies deswegen nicht möglich, weil er augenblicklich ausgeführt wird.

Der Befehl »MEMORY DUMP«

Die Hauptaufgabe eines Monitorprogramms ist es, Speicherbereiche anzuzeigen. Das geschieht natürlich hexadezimal.

Wir geben dem Monitor dazu die erste und die letzte anzuzeigende Adresse an. Geben Sie bitte folgenden Befehl ein (gefolgt von <RETURN>):

```
M 8180 81B5
```

Sie erhalten die in Bild 2 zu sehende Antwort des Monitors. Sollten Sie andere Werte erhalten, geben Sie zunächst

```
07F8 00
```

und dann wieder

```
M 8180 81B5
```

ein.

Die Anzeige bezeichnet man als »Memory Dump«, auf deutsch: »Speicherausdruck«. Ohne Erklärung kann man mit diesem hexadezimalen Zahlen- und Zeichen-Gewirr beim besten Willen nichts anfangen. Betrachten wir deshalb exemplarisch die erste Zeile (>8180 0A A9 ...).

Das Größer-Zeichen (>) am Anfang der Zeile ist erst an späterer Stelle von Bedeutung.

Die 4stellige Hex-Zahl danach, nämlich 8180, ist die sogenannte »Basisadresse« der Dump-Zeile. In dieser Zeile werden also die Inhalte der Speicherzellen ab 8180 angezeigt.

Diese Inhalte der Adressen stehen nach der Adresse in Form von 8 zweistelligen Hex-Zahlen. Nun benötigen wir besagte Basisadresse, um sagen zu können, welche Zahl der Inhalt welcher Speicherzelle ist. Die 8 Byte sind ganz einfach die Speicherinhalte ab der Basisadresse:

$0A ist der Inhalt von $8180, $A9 von $8181, $50 von $8182, und so weiter.

Nach dem Doppelpunkt stehen in der Reihenfolge der Bytes die Zeichen, zu denen die Bytes die ASCII-Codes, beziehungsweise die Commodore-Codes darstellen. Commodore-Codes deshalb, weil sich Commodore mit seinen vielen Sonderzeichen nur sehr begrenzt an die ASCII-Codierung hält. Wie wir wissen, hat jedes Zeichen einen Code, der aus einer Zahl von $00 bis $FF besteht. ASCII kennt aber nur Zeichen-Codes von $00 bis $7F. Bei unserem Computer benötigen wir deshalb einen eigenen Code, was uns aber im weiteren nicht stören soll.

Da Zeichen im Speicher als Zahlen abgelegt werden, zeigt unser Monitor zu jedem Byte den Code an, damit wir sofort sehen können, wo Texte stehen. In unserem Beispiel haben wir die Codes einiger Basic-Befehle erwischt.

Sie werden mit "M 8000 FFFF" auch noch andere Texte, die der Computer ausgibt, finden.

Bei der Code-Darstellung muß der Monitor noch ein kleines Problem lösen. Die Zeichen-Codes gibt es nämlich nicht nur für Buchstaben, Ziffern, Grafikzeichen und so weiter, sondern auch für sogenannte Steuerzeichen. Steuerzeichen sind im Gegensatz zu den »druckbaren Zeichen« diejenigen Codes, die kein Zeichen am Bildschirm ergeben, sondern eine Steuerfunktion hervorrufen. Mit <PRINT CHR$(65)> erhalten wir das druckbare Zeichen »A«, mit <PRINT

CHR$(147)> jedoch zum Beispiel das Steuerzeichen für »Bildschirm löschen«.

Da Steuerzeichen also nicht darstellbar sind, werden Sie vom Monitor in der Zeichen-Darstellung als Punkt ».« ausgegeben.

Es ist allerdings zu beachten, daß es den Punkt auch als richtiges Zeichen gibt. Dieser »echte« Punkt hat den ASCII-Code $2E oder dezimal 46 (probieren Sie in Basic PRINT CHR$(46)) und ist beim Memory Dump nicht von den Steuerzeichen zu unterscheiden. Im Zweifelsfall muß man also auf die Hex-Werte sehen, um erkennen zu können, ob es sich um einen wirklichen Punkt (Wert: $2E) oder ein Steuerzeichen (anderer Wert als $2E) handelt.

Am Anfang ist es schwer einzusehen, weshalb der M-Befehl so wichtig ist. Bald werden wir aber praktische Anwendungen besprechen, damit Sie einen ersten Eindruck von der Mächtigkeit dieses Monitor-Kommandos bekommen.

Zunächst müssen wir jedoch noch die Syntax des M-Befehls betrachten:

In unserem Beispiel (»M 8180 81B5«) haben wir die volle Syntax verwendet:

M Anfangsadresse Endadresse

Wenn wir die Endadresse weglassen, ergibt sich die Syntax »M Anfangsadresse«. Dann wird ab der angegebenen Anfangsadresse ein Memory Dump von 12 Zeilen Länge ausgegeben. Da in jeder Zeile 8 Adressen gedruckt werden, werden beim Weglassen der Endadresse also 12∗8 = 96 Byte ausgegeben. Die Adressen 0 bis 96 des C16-Speichers lassen Sie bitte über <M 0000> ausgeben, damit Sie einmal den Effekt gesehen haben.

Es gibt sogar noch eine dritte Syntax: <M> (ohne weitere Angaben danach).

Dann wird ab der letzten angezeigten Adresse 12 Zeilen lang gedruckt. Wenn Sie eben wie gefordert <M 0000> eingegeben haben, werden jetzt auf ein nachfolgendes <M> hin die Speicherzellen ab $0060 ausgegeben.

Der Befehl »M« ohne Parameter ist vor allem dann nützlich, wenn man das Dump zuvor durch <RUN/STOP> abgebrochen hatte.

In Bild 3 finden Sie die Kurzübersicht zum M-Befehl. Wir wollen nun zu ersten Anwendungen kommen.

Anwendung 1: Basic-Anfang und Basic-Ende

Wenn man ein Basic-Programm im Speicher hat, möchte man gerne wissen, wo es beginnt und wo es aufhört. Hat man diese Grenzen erst einmal ermittelt, kann man das Basic-Programm über »M« ausgeben lassen.

In $002B/$002C befindet sich die Anfangsadresse des Programms. Diese muß auf zwei Adressen ($002B UND $002C) aufgeteilt werden, da eine Adresse hexadezimal vierstellig ist, eine Speicherstelle aber nur zweistellige Hex-Zahlen aufnehmen kann. Der Computer bedient sich hier eines einfachen, aber wirkungsvollen Tricks: Die Speicherzelle mit der niedrigeren Adresse ($002B) erhält die beiden niederwertigen Stellen der vierstelligen Zahl, die andere Adresse ($002C) die beiden höherwertigen Stellen. Die niederwertigen Stellen nennt man »Low-Byte«, die höherwertigen »High-Byte«.

Ein Beispiel: Die Zahl $1AB5 soll in Low- und High-Byte aufgeteilt werden. Das Low-Byte besteht aus den niederwertigen (weiter rechts stehenden) Stellen und ist also $B5, das High-Byte entsprechend $1A.

In $002B liegt das Low-Byte, in $002C das High-Byte der Anfangsadresse des Basic-Programms im Speicher. Wir stellen jetzt diese Anfangsadresse fest.

Geben Sie dazu <M 002B 002C> ein. In 99 Prozent aller Fälle erhalten wir:

>002B 01 10 ...

Das Low-Byte ist somit $01, das High-Byte $10. Wie wir wissen, besteht das Low-Byte aus den beiden rechts stehenden Stellen, das High-Byte aus den beiden links stehenden. Richtig gestellt ist die Adresse also $1001. Wir hätten auch so vorgehen können, daß wir die Reihenfolge im Speicher (»01 10«) einfach umstellen und »1001« erhalten. Mit dem Basic-Befehl PRINT DEC ("1001") können wir unser »Forschungsergebnis« ins dezimale Format umwandeln lassen, wenn wir wollen.

Die Endadresse stellen wir auf gleiche Weise fest. Wir geben <M 002D 002E> ein und erhalten jetzt Werte, die von der Länge des aktuellen Basic-Programms abhängen. Jedenfalls nehmen wir die erste zweistellige Zahl und stellen die Ziffern der zweiten zweistelligen Zahl davor; das Ergebnis ist die Endadresse. War kein Programm im Speicher (zum Beispiel direkt nach dem Einschalten oder nach »NEW« der Fall), müßten Sie $1003 erhalten.

Anwendung 2: Basic-Programm ansehen

Laden Sie nun ein Basic-Programm ein und ermitteln Sie Anfangs- und Endadresse wie in Anwendung 1. Mit <M Anfangsadresse Endadresse> ist es dann möglich, das Basic-Programm im Speicher anzusehen. Warum dort wel-

```
Befehl:     X ("eXit")
Syntax:     X
Wirkung:    Rücksprung vom Monitor ins Basic
```

Bild 1. Kurzübersicht zum Befehl »X«

```
>8180 0A A9 50 D0 06 A9 6C D0  :
>8188 02 A9 5A 4C 94 04 45 4E  :
>8190 C4 46 4F D2 4E 45 58 D4  :
>8198 44 41 54 C1 49 4E 50 55  :
>81A0 54 A3 49 4E 50 55 D4 44  :
>81A8 49 CD 52 45 41 C4 4C 45  :
>81B0 D4 47 4F 54 CF 52 55 CE  :
```

Bild 2. Beispiel für die Anwendung des M-Befehls

```
Befehl:        M ("Memory dump")
Syntax:        M Anfangsadr. Endadr.
          oder M Anfangsadr.
          oder M
Wirkung:       Anzeige der Adresse und ihrer
               Speicherinhalte
```

Bild 3. Kurzübersicht zum Befehl »M«

```
Befehl:     >
Syntax:     > Adresse Byte1 Byte2 ByteX
Wirkung:    Die Bytes werden ab der
            Adresse abgelegt
```

Bild 4. Kurzübersicht zum >-Befehl

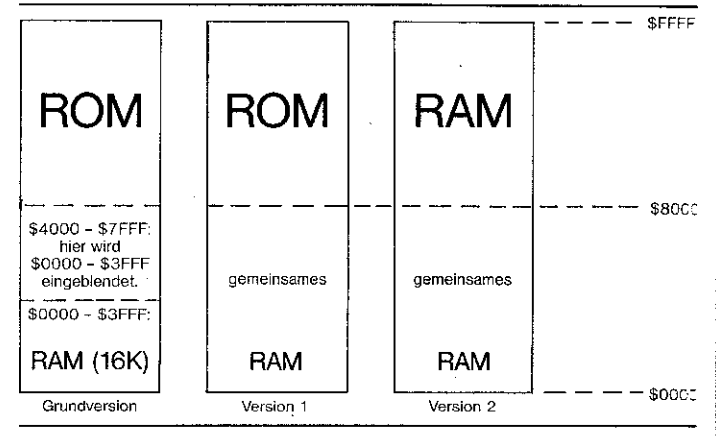


Bild 5. Bank-Switching. Im Bereich $0000 bis $7FFF ist immer RAM eingeblendet

che Werte stehen, besprechen wir am Ende des Kurses. Durch dieses Ansehen können Sie aber schon einen ersten Vorgeschmack bekommen.

Anwendung 3: Bildschirmspeicher betrachten

Diese Anwendung ist zwar unproduktiv, aber eine ganz gute Demonstration des M-Befehls.

Mit <M 0C00 0FE0> können wir den Bildschirmspeicher (das ist der Speicherbereich, in dem der Bildschirminhalt untergebracht ist) ausgeben. Wie Sie daraus ersehen können, wird auch der Bildschirm vom Computer wie ein normales RAM behandelt. Lediglich der Videoprozessor holt sich daraus seine Daten für das Aufbereiten des Bildes.

Sehen Sie sich das am besten einmal an.

Speicherzellen mittels Memory Dump ändern

Es ist interessant, die Speicherzellen mittels <M> anzusehen, und es vermittelt das Gefühl, hinter die Kulissen zu blicken.

Man verspürt aber auch das Bedürfnis, in das Geschehen einzugreifen, indem man die Inhalte der Speicherzellen ändert. Auch dafür kann man das Memory Dump einsetzen.

Wollen wir die angezeigten Werte ändern, müssen wir nur mit dem Cursor in die betreffende Dump-Zeile fahren, die hexadezimalen Werte mit unseren neuen Werten, die auch zweistellig hexadezimal sein müssen, überschreiben und <RETURN> drücken. Damit Sie es gleich ausprobieren können, geben Sie <M 1000> ein. Sie erkennen dort den Anfang des im Speicher befindlichen Basic-Programms (gegebenenfalls Monitor verlassen, Programm über Basic laden und dann den Monitor neu starten).

Gehen Sie mit dem Cursor in die erste Dump-Zeile und dort auf das erste Byte (nicht auf die Basisadresse, sondern auf die zweistellige Zahl danach!). Das erste Byte ist sicher »00«. Überschreiben Sie diesen Wert mit »5A« und drücken Sie dann <RETURN>.

Wie Sie sehen, wird die Änderung übernommen, denn in der ASCII-Darstellung verschwindet der Punkt, der das erste Byte repräsentierte, und es erscheint ein »Z« (das liegt am Wert $5A, dem ASCII-Code für »Z«). Verlassen Sie nun über <X> den Monitor und versuchen Sie, das Programm über <RUN> zu starten. Der Computer meldet einen »?SYNTAX ERROR«. Der Grund dafür ist die Änderung der Adresse $1000. Wenn wir sie wieder rückgängig machen (<M 1000>; erstes Byte mit »00« überschreiben; <RETURN> drücken), können wir das Programm auch wieder starten: <RUN> ergibt keinen »?SYNTAX ERROR« mehr.

Dieses Beispiel sollte nur eine kleine Demonstration sein, um das Prinzip zu verdeutlichen. Sie können in einer Zeile auch mehrere Bytes (im Extremfall alle) ändern; in jedem Fall dürfen Sie aber nur die Hex-Darstellung ändern (die ASCII-Darstellung können Sie zwar überschreiben, Ihre Änderung wird aber nicht in den Speicher übernommen).

Beim Ändern der Speicherinhalte wird die ASCII-Darstellung vom Computer nicht beachtet. Deshalb steht nach den Hex-Bytes der Doppelpunkt, der dem Monitor signalisiert, daß alles, was dem Doppelpunkt folgt, ignoriert werden soll.

Es ist also recht einfach, Speicherinhalte zu ändern. Man muß nur den entsprechenden Bereich ausgeben und dann die zu ändernden Bytes mit den gewünschten Werten überschreiben.

Das Größer-Zeichen (>) am Anfang einer Dump-Zeile ist nichts weiter als ein eigener Monitor-Befehl. Die Syntax lautet:

```
>Basisadresse Bytes
```

Die Basisadresse ist die Adresse, ab der die Speicherinhalte geändert werden sollen. Ab dieser Basisadresse werden dann die der Adresse folgenden Bytes abgelegt. Wir können nur ein Byte, aber auch mehrere Byte schreiben lassen.

Der Befehl >0C00 36 34 27 45 52 schreibt $36 in Adresse $0C00, $34 in Adresse $0C01 und so weiter.

Da diese Adressen der Anfang des Bildschirmspeichers sind, sehen Sie links oben am Bildschirm den Text »64'ER«.

Eine andere Anwendung wäre >FF19 00 (färbt den Rahmen schwarz).

Bild 4 ist die Kurzübersicht zum Befehl (>).

Den >-Befehl können wir auch zum sogenannten Bank-Switching einsetzen. Dazu müssen wir erst einmal klären, was Bank-Switching eigentlich ist.

Der Prozessor unseres C 16/116 und des Plus/4 kann nur 2^{16} = 65536 Adressen anwählen. In der Grundversion (von C 16/116) reicht das:

RAM	16 KByte
ROM	32 KByte
insgesamt	48 KByte
in Bytes (48 * 1024) =	49152

Der Speicher ist dann wie folgt aufgebaut: Von $0000 bis $4000 stehen die 16 KByte RAM, im Bereich $8000 bis $FFFF die 32 KByte ROM. Der Bereich $4000 bis $8000 ist genaugenommen ungenutzt, denn dort wird nur der RAM-Bereich $0000 bis $4000 eingeblendet. $4005 ist also nichts anderes als $0005. Diese Angaben gelten jedoch nur für die Grundversion von C 16 und C 116.

In diesem Fall kann man sagen, daß der Computer mit der Verwaltung des Speichers unterfordert ist, denn er könnte noch weitere 16 KByte adressieren.

Bei einer 64-KByte-Speichererweiterung sind jedoch 64 KByte RAM und 32 KByte ROM, also 96 KByte Speicher, zu adressieren (zum Beispiel im Plus/4).

Hier hat der Computer die Lösung des Bank-Switching parat. Es wird dabei zwischen zwei Adreßkonfigurationen umgeschaltet, in denen jeweils 64 KByte »sichtbar« sind:

Version 1 (ROM-Lösung)

$0000–$7FFF	RAM
$8000–$FFFF	ROM

In dieser Version wird nur ein Teil vom RAM (nur 32 KByte statt der gesamten 64 KByte) verfügbar gemacht; dafür aber ist das komplette ROM ab $8000 eingeblendet.

Version 2 (RAM-Lösung)

$0000–$FFFF	RAM

Das ROM wird ausgeblendet, dafür sind die gesamten 64 KByte RAM verfügbar.

Wir können jeweils nur eine Konfiguration einschalten. Verantwortlich hierfür ist die Adresse $07F8, die in beiden Versionen als RAM verfügbar ist.

Steht in $07F8 der Wert $00, haben wir Version 1 eingeschaltet, steht der Wert $80, so ist Version 2 gültig. So können wir mittels Umschalten auf jede Adresse im Speicher durch den Monitor zugreifen.

Version 1 ist übrigens der Einschaltzustand; wollen wir nur mit Version 1 arbeiten, müssen wir nie umschalten. Dies ist zum Beispiel dann der Fall, wenn Sie die Grundversion des C 16/116 haben, in der ja ein Bank-Switching nicht nötig ist.

Die Monitor-Befehle zum Umschalten sind:

```
>07F8 00: schaltet Version 1 ein
>07F8 80: schaltet Version 2 ein
```

In Bild 5 finden Sie noch eine grafische Darstellung des Bank-Switching.

Gezielte Suche nach Bytes

Wenn man den Speicher bearbeitet, möchte man vielleicht manchmal bestimmte Werte suchen. Dafür gibt es einen Suchbefehl.

Mit <H 8000 FFFF 49 4E D0> suchen wir im Bereich von $8000 bis $FFFF (natürlich in der aktuellen Speicherkonfiguration) nach der Bytefolge $49 $4E $D0.

Wir erhalten vom Monitor die Antwort:

```
8159 8165
```

Das bedeutet, daß die von uns gesuchte Bytefolge ab $8159 und ab $8165 im Speicher steht.

Wenn wir nach Zeichen suchen wollen, geht dies auch. Dazu müssen wir nach Anfangs- und Endadresse des zu durchsuchenden Bereichs ein Apostroph <'> eingeben und danach die Suchzeichen. Ein Beispiel:

```
H 8000 FFFF 'COMMODORE
```

sucht nach dem Text »COMMODORE« im Speicher von $8000 bis $FFFF. Wir erhalten als Antwort:

```
80CF E3A7
```

Falls Sie bei unseren beiden Beispielen zum H-Befehl nicht die genannten Adressen erhalten haben, haben Sie nicht die Speicherkonfiguration von Version 1 eingestellt, was Sie über <>07F8 00> leicht nachholen können.

Die Kurzübersicht zum H-Befehl ist in Bild 6 dargestellt.

An dieser Stelle soll Ihnen noch ein wichtiger Hinweis zum H-Befehl gegeben werden, der dringend erforderlich ist:

Wenn Sie den zu durchsuchenden Bereich so wählen, daß auch der Bereich $0200 bis $02FF geprüft wird, sind alle Werte von $0200 bis $025D, die der H-Befehl liefert, zu ignorieren. Der Grund ist, daß Werte aus diesem Bereich keine Aussagekraft haben.

Bei der C 16-Grundversion sind auch Werte von $4200 bis $425D zu vernachlässigen, da es sich nur um Kopien von $200 bis $25D handelt.

Die ASCII-Codes

Im Handbuch finden Sie zwar eine Tabelle der ASCII-Codes, aber diese führt nur den dezimalen Commodore-ASCII-Code auf.

Mit dem Programm in Listing 1 können Sie sich eine hexadezimale Tabelle ausgeben lassen. Diese Tabelle stellt bei den ASCII-Codes $00 bis $1F und $80 bis $9F kein Zeichen dar, da es sich hierbei um Steuerzeichen handelt.

Bei $20, $A0 und $E0 steht auch kein Zeichen, denn diese Codes stehen für das Leerzeichen (»Space«).

Wenn Sie bei der Arbeit mit dem Monitor den Commodore-ASCII-Code für ein bestimmtes Zeichen wissen wollen, nehmen Sie die Tabelle zur Hand und suchen das Zeichen. Dann finden Sie die erste hexadezimale Stelle in der waagrechten Spaltennumerierung von 0 bis F, die zweite Hex-Stelle in der senkrechten Numerierung der Zeilen (auch von 0 bis F). Nehmen wir als Beispiel das Ausrufezeichen »!«: die erste Stelle ist 2, die zweite ist 1. Der hexadezimale ASCII-Code ist also $21.

So läßt man die Umrechnung über Basic ausführen:

```
PRINT RIGHT$(HEX$(ASC("A")),2)
```

ergibt den hexadezimalen ASCII-Code des Buchstabens »A« als zweistellige Zahl; andere ASCII-Codes erhalten Sie, wenn Sie statt »A« andere Zeichen einsetzen.

Bei manchen Zeichen geht die Umrechnung ganz einfach. Der ASCII-Code einer Ziffer hat als linke Hex-Ziffer die »3« und als rechte Stelle die Ziffer selbst; der ASCII-Code von 5 ist also $35.

Ähnlich leicht errechnet sich der ASCII-Code der Zeichen, die über <SHIFT> und eine Zahlentaste erreicht werden (zum Beispiel das Ausrufezeichen <SHIFT 1>, Und-Symbol <SHIFT 6>); man nimmt die »2« als linke Hex-Ziffer und als rechte Stelle verwendet man die Zahlentaste, die man außer <SHIFT> drücken muß, um das Zeichen zu erhalten.

Das raffinierteste Umrechnungsverfahren aber bedient sich nur des Monitors.

Wir rechnen damit den Text »BEISPIEL« in den ASCII-Code um. Dazu geben wir zunächst

```
H 0 0 'BEISPIEL
```

ein. Dieser Suchbefehl bringt keine Adresse auf den Bildschirm, das soll er aber auch gar nicht. Nun steht im Suchpuffer des Monitors dieser Text im ASCII-Code. Folgender Befehl

```
Befehl:      H ("Hunt")
Syntax:      H Anfangsadr. Endadr. Byte1 ... ByteX
        oder H Anfangsadr. Endadr. 'Text
Wirkung:     Suchen der Bytes oder des Textes im
             durch Anfangsadr. und Endadr.
             eingegrenzten Bereich
```

Bild 6. Kurzübersicht zum H-Befehl

ASCII-Code	Wirkung
$05	Cursorfarbe weiß
$08	SHIFT C = verbieten
$09	SHIFT C = wieder zulassen
$0D	RETURN (Cursor an Anfang der nächsten Zeile)
$0E	Klein/Großschrift ein
$11	Cursor eine Zeile nach unten
$12	Reversschrift ein
$13	Cursor in linke obere Ecke
$14	letztes Zeichen löschen
$1B	ESC
$1C	Cursorfarbe rot
$1D	Cursor eine Spalte nach rechts
$1E	Cursorfarbe grün
$1F	Cursorfarbe blau
$82	FLASH ON (Blinken ein)
$83	FLASH OFF (Blinken aus)
$8D	SHIFT RETURN (wie $0D)
$8E	Groß/Grafikschrift ein
$90	Cursorfarbe schwarz
$91	Cursor eine Zeile nach oben
$92	Reversschrift aus
$93	Bildschirm löschen
$94	ein Zeichen einfügen (INST)
$9C	Cursorfarbe purpur
$9D	Cursor eine Spalte nach links
$9E	Cursorfarbe gelb
$9F	Cursorfarbe türkis

Bild 7. Die Steuerzeichen und Ihre Hex-Codes

```
Befehl:      T ("Transfer")
Syntax:      T Anfangsadr. Endadr. Zieladr.
Wirkung:     verschiebt den Bereich von
             Anfangsadr.
             bis Endadr. an die Zieladr.
Beispiel:    T 8000 8300 C00 verschiebt einen Teil
             des ROMs in den Bildschirmspeicher;
             am Bildschirm erscheint ein
             Zeichengewirr.
```

Bild 8. Kurzübersicht zum Befehl »T«

```
Befehl:      C ("Compare")
Syntax:      C Anfangsadr. Endadr. Vergleichsadr.
Wirkung:     vergleicht den Bereich von
             Anfangsadr. bis Endadr. mit dem
             Bereich ab Vergleichsadr.;
             unterschiedliche Adressen werden
             ausgegeben.
Beispiel:    C 80CF 80D9 E3A7
```

Bild 9. Kurzübersicht zum C-Befehl

```
100 REM ********************
110 REM *                  *
120 REM *  ASCII - TABELLE *
130 REM *                  *
140 REM ********************
150 REM *                  *
160 REM *  FLORIAN MUELLER *
170 REM *                  *
180 REM ********************
190 :
200 DO : INPUT "DRUCKER (D) ODER BILDSCHIRM (B)"
    ;A$: LOOP WHILE A$<>"D" AND A$<>"B"
210 IF A$="B" THEN 250
220 INPUT "GERAETENUMMER";G: INPUT "SEKUNDAERADR
    ESSE";S
230 OPEN 4,G,S: CMD 4
240 :
250 LE$=CHR$(32): REM LEERZEICHEN
260 Z$="0123456789ABCDEF": REM ZIFFERN FUER HEX-
    ZAHLEN
270 PRINT SPC(2);
280 :
290 FOR J = 1 TO 16
300 : PRINT MID$(Z$,J,1);LE$;
310 NEXT J
320 PRINT
330 :
340 FOR J = 0 TO 15
350 : PRINT MID$(Z$,J+1,1);SPC(5)
360 : FOR K = 2 TO 15: IF K>7 AND K<10 THEN 390
370 : : PRINT CHR$(J+(K*16));LE$;
380 : : IF K=7 THEN PRINT LE$;LE$;LE$;LE$;
390 : NEXT K
400 : PRINT
410 NEXT J
420 :
430 IF A$="D" THEN PRINT# 4: CLOSE 4
```

Listing 1. Erstellen einer ASCII-Tabelle

gibt diesen Suchpuffer aus, wo wir die ASCII-Codes des Textes dem Memory Dump entnehmen können:

```
M 0 25D
```

Diese Art von Umrechnung ist immer noch die beste, schnellste und komfortabelste.

Zurück zu den Basic-Befehlen für ASCII-Codes. Mit folgendem Befehl läßt man das zu einem ASCII-Code gehörende Zeichen ausgeben:

```
PRINT CHR$(DEC("hex.ASCII-Code"))
```

Mit <PRINT CHR$(DEC("41"))> erhalten wir zum Beispiel das Zeichen »A«.

Probieren Sie aber doch einmal die ASCII-Codes $13, $93, $1F, $90 oder $05 aus. Wie Sie sehen, ergeben diese kein Zeichen, sondern positionieren den Cursor, ändern die Zeichenfarbe oder ähnliches. Dies sind sogenannte Steuerzeichen. In Bild 7 finden Sie alle C 16-Steuerzeichen mit ihren Commodore-ASCII-Codes.

Ein interessantes Steuerzeichen ist »ESC« (27); dazu lesen Sie bitte im Handbuch nach.

Wir wollen nun weitere Monitor-Kommandos kennenlernen, die einfach zu handhaben sind. Alle wichtigen Informationen über die Syntax und so weiter finden Sie in den Kurzübersichten.

Der Befehl <T> dient zum Verschieben von Speicherbereichen (Bild 8). Dieser Verschiebevorgang wird fortlaufend überprüft und Adressen, an denen das Verschieben nicht ordnungsgemäß durchgeführt werden konnte, werden ausgegeben.

Der Vergleich von Speicherbereichen ist mit <C> möglich (Bild 9), ganze Speicherbereiche mit einem Wert anfüllen kann man über <F> (Bild 10).

Um Speicherbereiche zu laden, speichern oder zu verifi-

zieren, gibt es die Befehle <S>, <L> und <V> (Bilder 11 bis 13), die den Basic-Befehlen »LOAD«, »SAVE« und »VERIFY« entsprechen, sich aber nicht auf das im Speicher befindliche Basic-Programm, sondern auf einen Teil des Speichers, den Sie selbst bestimmen, beziehen.

Bei <S> wird der Inhalt der Endadresse nicht mitgespeichert; »S "NAME",1000,2000« speichert also von $1000 bis $1FFF, nicht bis $2000. Gegebenenfalls müssen Sie also die Endadresse um 1 erhöhen.

Bearbeiten Sie ein Basic-Programm, ist es besser, die entsprechenden Basic-Befehle (»LOAD«, »SAVE«, »VERIFY«) heranzuziehen. Sie können aber auch die Monitor-Kommandos verwenden, wenn Sie Anfangs- und Endadresse des Basic-Programms kennen. Die Monitor-Befehle sind lediglich dann unausweichlich, wenn Sie nur bestimmte Teile des Speichers speichern oder laden wollen.

Wir wollen uns noch ein wenig mit der Ein-/Ausgabe des TEDMON befassen. Zunächst klären wir die beiden unterschiedlichen Ladevorgänge. Wenn Sie ein Programm über <L "NAME", Gerät> laden, wird es an die Adresse, ab der es gespeichert wurde, geladen.

Wenn ein Programm gespeichert wird, kommen auf das Speichermedium nicht nur die Daten des Programms, sondern auch die Anfangsadresse. Auf Diskette geben die ersten zwei Bytes des Programms die Anfangsadresse an (1. Byte: Low-Byte; 2. Byte: High-Byte).

Listing 2 (»FILE-INFO«) ermöglicht Ihnen, Anfangs- und Endadresse eines Programms zu ermitteln, das auf einer Diskette gespeichert ist.

Nach Eingabe des Filenamens (<$> bewirkt Ausgabe des Disk-Inhaltsverzeichnisses) stellt das Programm die Anfangsadresse und die Anzahl der Diskettenblöcke fest. Daraus läßt sich (aus der Kapazität eines Diskettenblocks) die maximale Endadresse des untersuchten Programms ermitteln. Die genaue Endadresse wird in einem etwas längeren Vorgang festgestellt; diese können Sie dann direkt als Parameter für den S-Befehl einsetzen. Am besten erstellen Sie sich einen Katalog aller Programme, die Sie mit dem Monitor bearbeiten wollen, wobei die optionale Druckerausgabe sicher eine Hilfe ist.

Insbesondere, wenn Sie später mit Maschinenprogrammen arbeiten wollen, werden Sie auf Listing 2, das »FILE-INFO«, nicht verzichten wollen.

TEDMON und hochauflösende Grafik

Eine Anwendung des S-Befehls ist das Speichern der hochauflösenden Grafik.

Mit

```
S "GRAFIK",Gerät,2000,4000
```

können Sie die hochauflösende Grafik speichern.

Mit

```
L "GRAFIK",Gerät
```

wird sie wieder geladen.

Druckerausgabe mit dem TEDMON

Leider bietet der TEDMON keinen Befehl, um die Ausgabe vom Bildschirm auf den Drucker umzulenken. Mit einem Basic-Befehl ist das aber möglich. Geben Sie in Basic

```
OPEN 4,4:CMD 4:MONITOR
```

ein, wird vor dem Start des Monitors die Ausgabe auf den Drucker umgelenkt. Dies geht soweit, daß sogar die Einschaltmeldung des TEDMON auf den Drucker geleitet wird. Sie geben aber ungeachtet dessen die Monitor-Kommandos wie üblich ein. Alle Ausgaben des Monitors (Memory Dumps etc.) kommen nun auf den Drucker.

Wenn Sie fertig sind, genügt es nicht, beide Geräte einfach auszuschalten, sondern Sie müssen den Monitor über <X> verlassen und in Basic dann

```
Befehl:    F ("Fill")
Syntax:    F Anfangsadr. Endadr. Füllbyte
Wirkung:   füllt den Bereich von Anfangsadr. bis
           Endadr. mit dem Füllbyte
Beispiel:  F C00 F00 01
           füllt den Bildschirm(speicher) mit
           "A"
```

Bild 10. Kurzübersicht zum Befehl »F«

```
Befehl:    S ("Save")
Syntax:    S "Filename",Gerätenummer,
           Anfangsadr. Endadr.
Wirkung:   unter dem Filenamen in Anführungs-
           zeichen wird der Bereich von Anfangs-
           adr. bis Endadr. auf das Gerät, das
           durch "Gerätenummer" angewählt wird,
           abgespeichert.
```

Bild 11. Kurzübersicht zum Befehl »S«

```
100 REM ********************************
110 REM *                              *
120 REM *        ANFANGSADRESSE        *
130 REM *                              *
140 REM *        UND ENDADRESSE        *
150 REM *                              *
160 REM *   EINES DISKETTENPROGRAMMS   *
170 REM *                              *
180 REM *          ERMITTELN.          *
190 REM *                              *
200 REM ********************************
210 REM *                              *
220 REM *  1985/86 BY FLORIAN MUELLER *
230 REM *                              *
240 REM ********************************
250 :
260 SCNCLR : W$="#####"
270 PRINT CHR$(27)"O": PRINT "BESTIMMUNG VON ANF
    ANGS- UND ENDADRESSE" CHR$(13)
280 INPUT "FILENAME";N$: IF N$="$" THEN PRINT :
    DIRECTORY : PRINT : GOTO 270
290 OPEN 15,8,15,"I0"
300 IF DS<>0 THEN 330
310 OPEN 1,8,3,N$+",P,R"
320 IF DS=0 THEN 400
330 PRINT : PRINT CHR$(15)"DISKFEHLER: ";DS$: PR
    INT
340 CLOSE 1: CLOSE 15: GOTO 280
350 :
360 REM **************************
370 REM * INFORMATIONEN AUSGEBEN *
380 REM **************************
390 :
400 SCNCLR : PRINT : PRINT "INFORMATIONEN FUER "
     CHR$(34);N$;CHR$(34)
410 PRINT
420 :
430 REM ******************
440 REM * ANFANGSADRESSE *
450 REM ******************
460 :
470 PRINT : PRINT "ANFANGSADRESSE:",
480 GET #1,A$: GET #1,B$
490 IF A$="" THEN A$=CHR$(0)
500 IF B$="" THEN B$=CHR$(0)
510 A=ASC(A$)+256*ASC(B$)
520 PRINT USING W$;A;: PRINT " = $";HEX$(A)
530 CLOSE 1
540 :
550 REM ***********************
560 REM * ANZAHL DER BLOECKE *
570 REM ***********************
580 :
590 PRINT : PRINT "DISKBLOECKE:",
600 OPEN 1,8,0,"$:"+N$
610 GET #1,A$,B$
620 GET #1,A$,B$
630 GET #1,A$,B$
640 GET #1,B$: IF ST<>0 THEN 830
650 IF B$<>CHR$(34) THEN 640
660 DO : GET #1,B$: LOOP WHILE B$<>CHR$(34)
670 DO : GET #1,B$: LOOP WHILE B$=CHR$(32)
```

```
Befehl:         L ("Load")
Syntax:         L "Filename",Gerät,Ladeadresse
           oder L "Filename",Gerät
           oder L "Filename"
Wirkung:        Das Programm »Filename« wird vom Gerät
                an die Ladeadresse geladen. Fällt die
                Ladeadresse weg, wird das Programm in
                den Bereich geladen, aus dem es auch
                abgespeichert wurde. Gibt man das
                Gerät nicht an, wird $01 (Datasette)
                angenommen.
                Durch die Angabe einer Ladeadresse
                kann man das Programm beim Laden in
                einen anderen Speicherbereich
                schreiben als den, in dem es vor dem
                Speichern stand.
```

Bild 12. Kurzübersicht zum L-Befehl

```
680 DO : GET #1,B$: LOOP WHILE B$<>""
690 GET #1,A$,B$,A$,B$
700 IF A$="" THEN A$=CHR$(0)
710 IF B$="" THEN B$=CHR$(0)
720 B=ASC(A$)+256*ASC(B$)
730 PRINT USING W$;B;: PRINT " = $";HEX$(B)
740 CLOSE 1
750 :
760 REM ***********************
770 REM * MAXIMALE ENDADRESSE *
780 REM ***********************
790 :
800 PRINT : PRINT "MAX.ENDADRESSE:",
810 E=A+254*B-2
820 PRINT USING W$;E;: PRINT " = $";HEX$(E)
830 CLOSE 1: CLOSE 15
840 PRINT : PRINT CHR$(18)"TASTE DRUECKEN"
850 PRINT : PRINT "(G = GENAUE ENDADR./ D = DRUC
    KERAUSGABE)";
860 GET KEY GK$: IF GK$="D" THEN EE=E: M=1: GOTO
     1070: ELSE IF GK$<>"G" THEN RUN
870 :
880 REM ********************
890 REM * GENAUE ENDADRESSE *
900 REM ********************
910 :
920 M=0
930 PRINT CHR$(27)"D";CHR$(145) CHR$(145)"ENDADR
    ESSE FUER 'S':";
940 EE=A: OPEN 1,8,0,N$
950 DO UNTIL ST AND 64
960 : GET #1,A$: EE=EE+1
970 LOOP : EE=EE-1
980 PRINT USING W$;EE;: PRINT " = $";HEX$(EE)
990 CLOSE 1
1000 PRINT : PRINT CHR$(18)"TASTE DRUECKEN (D =
     DRUCKERAUSGABE)";CHR$(146);
1010 GET KEY GK$: IF GK$<>"D" THEN RUN
1020 :
1030 REM *****************
1040 REM * DRUCKERAUSGABE *
1050 REM *****************
1060 :
1070 V$="##############################": REM 30
     * #
1080 PRINT CHR$(27)"D"
1090 OPEN 4,3
1100 PRINT#4,USING V$;"FILENAME:";: PRINT#4,N$
1110 PRINT#4,USING V$;"ANFANGSADRESSE:";: PRINT#
     4,USING W$;A;
1120 PRINT#4," = $";HEX$(A)
1130 PRINT#4,USING V$;"ENDADRESSE:";: PRINT#4,US
     ING W$;E;: PRINT#4," = $";HEX$(EE);
1140 IF M=1 THEN PRINT#4," (MAXIMAL)": ELSE PRIN
     T#4," (FUER 'S'-BEFEHL)"
1150 PRINT#4,USING V$;"LAENGE IN DISK-BLOECKEN:"
     ;: PRINT#4,USING W$;B;
1160 PRINT#4," = $";HEX$(B)
1170 CLOSE 4
1180 IF M=1 THEN 840: ELSE GOTO 1000
```

Listing 2. »File-Info« für C16, C116 und Plus/4

PRINT#4:CLOSE4
eingeben. Wenn Sie das unterlassen, ist es gut möglich, daß der Drucker eine auszugebende Zeile nicht aufs Papier bringt, sondern »verschluckt«.

Jetzt haben wir alle Befehle, die zum Bearbeiten von Basic-Programmen mit dem Monitor erforderlich sind, besprochen. Manche Befehle werden wir für unsere Anwendungen auf Basic-Programme, zu denen wir nun kommen, noch gar nicht benötigen. Ich habe diese Kommandos dennoch vorgestellt, denn Sie könnten durchaus in die Lage kommen, auf diese zurückgreifen zu müssen.

Basic-Programme durchleuchten

Als eine der ersten Anwendungen des M-Befehls haben wir uns ein Basic-Programm angesehen. Dabei blieben noch viele Fragen offen (zum Beispiel warum man kein Befehlswort wie »PRINT« oder »GOTO« sehen konnte), die nun nach und nach beantwortet werden können. Dazu analysieren wir das Memory Dump eines Beispiel-Programms, an dem wir später auch Manipulationen durchführen wollen. Dieses Beispiel-Programm ist Listing 3. Das Memory Dump lassen Sie nun wie beim M-Befehl besprochen ausgeben (Anwendung 2 des M-Befehls).

Sie erhalten dann den Ausdruck, den Sie auch in Bild 14 sehen können.

Listing 3 müssen Sie übrigens exakt so eintippen, wie es ausgedruckt ist; sie dürfen (außer direkt nach der Zeilennummer) kein Leerzeichen auslassen!

```
100 REM "PROGRAMM VERSUCHSKANINCHEN"
110 SCNCLR : PRINT "(C) 64'ER"
120 REM ****************************
130 COLOR 4,1
140 END
```

Listing 3. Kleines »Versuchskaninchen«

Wir besprechen nun anhand des »Versuchskaninchen«-Programms den Aufbau eines Basic-Programms im Speicher.

Die Linkpointer

Die ersten Bytes des Programms enthalten den sogenannten »Linkpointer« der ersten Programmzeile. Der Begriff »Linkpointer« heißt auf deutsch etwa »Verbindungszeiger«. Der Computer benötigt den Linkpointer, um zu wissen, wo im Speicher die nächste Programmzeile steht. Am Anfang jeder Zeile steht im Speicher der Linkpointer auf die nächste. In unserem Fall – der Linkpointer ist immer im Low-High-Format abgelegt – beginnt also die nächste Zeile bei $1024, denn am Anfang steht »24 10«.

Die Zeilennummer

Der Computer muß sich auch die Zeilennummer merken. Diese legt er ebenfalls im Low-High-Format ab. Die Zeilennummer steht unmittelbar nach dem Linkpointer, also im 3. und 4. Byte jeder Zeile. In unserem Fall ist die Zeilennummer durch die Bytes »64 00« bestimmt ($0064 = 100).

Die Token

Nach der Zeilennummer finden wir im Listing des Programms ein Leerzeichen. Das nächste Byte im Speicher (»8F«, siehe Memory Dump) ist aber kein Leerzeichen.

Beim LISTen eines Programms fügt der Computer automatisch nach der Zeilennummer genau ein Leerzeichen ein; bei der Eingabe aber ignoriert er alle Leerzeichen zwischen Zeilennummer und Befehl. Es ist also völlig egal, ob Sie

```
100        REM "PROGRAMM VERSUCHSKANINCHEN"
```

oder

```
100REM"PROGRAMM VERSUCHSKANINCHEN"
```

eingeben.

Später gibt er der Übersichtlichkeit halber von sich aus eines nach der Zeilennummer aus.

Im Speicher ist also das nächste Byte nicht das Leerzeichen, sondern der REM-Befehl.

Nun finden wir aber nach der Zeilennummer nicht die ASCII-Codes des Textes »REM«, sondern nach der Zeilennummer (»64 00«) nur das Byte »8F«).

Der Grund ist, daß Befehlswörter nicht im ASCII-Format gespeichert werden, sondern schon bei der Eingabe eine Codierung in Token erfolgt. Jedes Befehlswort wird durch ein Byte (im Beispiel »8F« für REM) repräsentiert. Dies spart Speicherplatz und erhöht die Verarbeitungsgeschwindigkeit, weil jetzt anstatt mehrerer Bytes nur 1 Byte ($8F) vom Computer bearbeitet werden muß.

Token erkennt der Computer daran, daß sie größer als $80 sind.

Es ist noch anzumerken, daß der Computer nur Befehlswörter außerhalb von Anführungszeichen in Token umwandelt.

Die Endmarkierung einer Zeile

Nach dem REM-Token (»8F«) finden wir die ASCII-Codes des Textes nach REM. Im ASCII-Code merkt sich der Computer einen Großteil des Programms, nämlich:
- Texte in Anführungszeichen
- Texte nach einem REM-Befehl
- Texte, die nicht Befehlswörter (also zum Beispiel Variablen) sind



In der Dump-Zeile mit der Basisadresse $1021 steht am Anfang »4E 22«. Das sind die ASCII-Codes für »N« und das Anführungszeichen »"«, also der Schluß des Textes nach dem REM-Befehl in Zeile 100. Da an dieser Stelle die Zeile 100 aufhört, folgt eine Endmarkierung in Form eines $00-Bytes. Am Ende jeder Zeile steht $00. Darauf folgt dann die nächste Zeile in der gewohnten Reihenfolge »Linkpointer – Zeilennummer – Befehl – Endmarkierung«. Nach der letzten Zeile steht dann ein $00-Byte als Endmarkierung der Zeile ($1068).

Dann folgen aber noch zwei weitere $00-Bytes ($1069/$1070), die als Endmarkierung des gesamten Programms dienen. Im Memory Dump stehen nach $1070 zwar

```
Befehl:        V ("Verify")
Syntax und Wirkung siehe L-Befehl (Bild 12); bei V wird
das Programm nicht geladen, sondern wie beim Befehl
"VERIFY" in Basic verglichen.
```

Bild 13. Kurzübersicht zum V-Befehl

```
>1001 24 10 64 00 8F 20 22 14  :
>1009 0D 93 12 4B 4F 50 46 5A  :
>1011 45 49 4C 45 92 20 2A 2A  :
>1019 2A 2A 0D 0D 1B 54 2A 2A  :
>1021 2A 8D 00 37 10 6E 00 E8  :
>1029 3A 99 22 28 43 29 20 36  :
>1031 34 27 45 52 22 00 5A 10  :
>1039 78 00 8F 20 1B 44 45 53  :
>1041 43 2D 44 20 4C 4F 45 53  :
>1049 43 48 54 20 44 49 45 20  :
>1051 5A 45 49 4C 45 2E 2E 2E  :
>1059 00 63 10 82 00 E7 34 2C  :
>1061 31 00 69 10 8C 00 80 00  :
>1069 00 00 00 00 00 00 00 00  :
```

Bild 14. Memory Dump des Programms »Versuchskaninchen«

```
>1001 24 10 64 00 8F 20 22 50  :
>1009 52 4F 47 52 41 4D 4D 20  :
>1011 56 45 52 53 55 43 48 53  :
>1019 4B 41 4E 49 4E 43 48 45  :
>1021 4E 22 00 37 10 6E 00 E8  :
>1029 3A 99 22 28 43 29 20 36  :
>1031 34 27 45 52 22 00 5A 10  :
>1039 78 00 8F 20 2A 2A 2A 2A  :
>1041 2A 2A 2A 2A 2A 2A 2A 2A  :
>1049 2A 2A 2A 2A 2A 2A 2A 2A  :
>1051 2A 2A 2A 2A 2A 2A 2A 2A  :
>1059 00 63 10 82 00 E7 34 2C  :
>1061 31 00 69 10 8C 00 80 00  :
>1069 00 00 00 00 00 00 00 00  :
```

Bild 15. »Versuchskaninchen« mit Steuercodes

```
>1001 24 10 64 00 8F 0D 12 50  :
>1009 52 4F 47 52 41 4D 4D 20  :
>1011 56 45 52 53 55 43 48 53  :
>1019 4B 41 4E 49 4E 43 48 45  :
>1021 4E 0D 00 37 10 6E 00 E8  :
>1029 3A 99 22 28 43 29 20 36  :
>1031 34 27 45 52 22 00 5A 10  :
>1039 78 00 8F 20 1F 42 4C 41  :
>1041 55 13 55 4E 44 20 57 49  :
>1049 45 44 45 52 20 4C 49 4E  :
>1051 4B 53 20 4F 42 45 4E 2E  :
>1059 00 63 10 82 00 E7 34 2C  :
>1061 31 00 69 10 8C 00 80 00  :
>1069 00 00 00 00 00 00 00 00  :
```

Bild 16. Programm mit Codes, die größer als $7F sind und mit ESC-Befehlen

noch mehrere Nullen, diese gehören aber nicht mehr zum Programm, sondern stehen nur zufällig dort.

Nachdem wir nun den Aufbau besprochen haben, sollten Sie in der Lage sein, jedes einzelne Byte im Memory Dump des Beispielprogramms zu erklären; die Token müssen Sie natürlich nicht auswendig wissen.

Wir wenden uns den ersten Manipulationen an unserem »Versuchskaninchen« zu. Speichern Sie bitte spätestens jetzt dieses Programm.

Linkpointer-Manipulationen

Die Linkpointer werden vom Computer nach jedem Laden eines Programms neu berechnet; deshalb kann man zwar die Linkpointer trickreich manipulieren und das Programm abspeichern, nach dem Neuladen aber ist die ganze Mühe umsonst. Aus diesem Grund sind die Effekte, die wir nun hervorrufen werden, lediglich eine Demonstration, von einem echten LIST-Schutz kann man nicht sprechen.

Als erstes wollen wir den Linkpointer von Zeile 100 auf sich selbst stellen, so daß beim LISTen nur immer Zeile 100 gelistet wird, der Rest des Programms aber nicht.

Der erste Linkpointer steht bei $1001 und zeigt noch auf $1024. Geben Sie <M 1001> ein und ändern Sie die beiden ersten Bytes auf <01 10> ($01: Low-Byte von $1001, $10: High-Byte), oder geben Sie einfach <>1001 01 10> ein.

»Vermurkste« Basic-Programme

Um das Zerstörungswerk anzusehen, verlassen wir den Monitor und geben <LIST> ein. Die Zeile 100 wird ununterbrochen gelistet, bis wir <RUN/STOP> drücken.

Sicherlich ist dies eine nette LISTschutz-Variante, doch können wir sie nicht einsetzen, da beim Neuladen eines Programms von Diskette oder Datasette derartige LISTschutz-Vorrichtungen entfernt werden. Der Computer bedient sich dabei einer Routine, die Sie über

```
SYS DEC("8818")
```

aufrufen können. Nach diesem Befehl ist das Programm wieder »repariert«, da die Linkpointer neu berechnet wurden.

Das soll uns aber nicht davon abhalten, weiter mit den Linkpointern zu experimentieren. Als nächstes wollen wir die Endmarkierung des Programms so ändern, daß nach dem LISTen der letzten Zeile wieder die erste Zeile geLISTet wird.

Der Linkpointer der letzten Zeile steht bei $1063 und zeigt auf die Endmarkierung ($1069). Wir ändern also das »69 10« in »01 10«, und schon haben wir ein weiteres Endlos-Listing, bei dem allerdings jede Programmzeile gelistet wird. Auch hier hilft <RUN/STOP> und dann ein

```
SYS DEC("8818")
```

weiter.

Der letzte Linkpointer-Trick ist, daß wir die Linkpointer so stellen, daß bestimmte Zeilen für den LIST-Befehl nicht vorhanden sind. Solche Zeilen dürfen aber auch nicht angesprungen werden.

Folgendermaßen sind die Linkpointer in »Versuchskaninchen« gestellt:

$1001 (Zeile 100) zeigt auf $1024,
$1024 (Zeile 110) zeigt auf $1037,
$1037 (Zeile 120) zeigt auf $105A,
$105A (Zeile 130) zeigt auf $1063,
$1063 (Zeile 140) zeigt auf $1069,
bei $1069 stehen Nullbytes als Endmarkierung.

Wir richten den ersten Linkpointer auf den letzten:

```
>1001 63 10
```

Damit wird auf $1063 (Zeile 140) gestellt.

Die Eingabe von <LIST> zeigt, daß die Zeilen 110,120 und 130 beim LISTen verschluckt werden, obwohl sie noch vorhanden sind, wie <SYS DEC("8818")> beweist.

Zugegeben: Ein wenig frustrierend ist es schon, wenn man so raffinierte Sachen mit den Linkpointern anstellt, und dann durch Speichern und Neuladen, beziehungsweise <SYS DEC("8818")> alles wieder weg ist.

Die Manipulationen, die wir nun durchführen werden, sind aber wesentlich »wetterfester«, denn man kann sie der Nachwelt durch Speichern erhalten.

Manipulationen an Zeilennummer

Wie wir wissen, kann man mit zwei Byte alle (ganzen) Zahlen von 0 bis 65535 darstellen. Dennoch läßt der Computer, der Basic-Zeilennummern ja in diesem Low-High-Format speichert, nur Zeilennummern bis 63999 zu. Mit dem Monitor sind wir aber nicht auf den normalen Basic-Eingabe-Modus angewiesen und können bestehende Zeilennummern so ändern, daß wir auch Zeilennummern von 64000 bis 65535 verwenden können.

Geben Sie folgende Monitor-Anweisungen ein (selbstverständlich ohne die dahinterstehenden Kommentare in Klammern), und schon hat unser »Versuchskaninchen« die Zeilennummern 64000 bis 64040 anstelle von 100 bis 140:

```
>1003  00 FA         (64000 statt 100)
>1026  0A FA         (64010 statt 110)
>1039  14 FA         (64020 statt 120)
>105C  1E FA         (64030 statt 130)
>1065  28 FA         (64040 statt 140)
```

Lassen Sie das Programm jetzt LISTen, Sie werden erstaunt sein!

Zwei Anmerkungen.

Erstens: Das Programm läuft nur dann korrekt, wenn Zeilen mit unerlaubt großer Zeilennummer nicht angesprungen werden, denn den Befehlen <GOTO>, <GOSUB> oder <RUN> können wir die Zeilennummern über 64000 nicht beibringen.

Zweitens: Mit einem einfachen RENUMBER-Befehl können wir auch diese Manipulation rückgängig machen; <RENUMBER 100,10> stellt den ursprünglichen Zustand wieder her.

Wir können auch allen Zeilen die Zeilennummer 100 »verpassen«:

```
>1003 64 00
>1026 64 00
>1039 64 00
>105C 64 00
>1065 64 00
```

Es gelten die gleichen Hinweise wie bei den »überdimensionalen« Zeilennummern.

Unsere nächste Anwendung ist eine echte Hilfe beim Programmieren.

Das Basic 3.5 unseres C 16/116 ist zwar wirklich umfangreich, aber der FIND-Befehl zum Suchen von Texten innerhalb von Programmen fehlt leider. Mit dem Monitor können wir einen solchen unter einigermaßen vertretbarem Aufwand ersetzen.

Wir suchen jetzt als Beispiel nach dem Text »64« im Programm »Versuchskaninchen«.

I. Bestimmung von Anfangs- und Endadresse des Basic-Programms

Diesen Schritt kennen wir seit dem M-Befehl. Bei »Versuchskaninchen« ist $1001 die Anfangsadresse und $106B die Endadresse.

II. Suchen nach dem Text im Speicher

```
H 1001 106B '64
```

sucht nun nach dem Text »64«. Wir erhalten die Adresse $1030 als Ergebnis.

Befehl	Syntax	Wirkung
eXit	X	Rückkehr in Basic
Memory dump	M Start Ende	Anzeige von Speicherinhalten
>	> Adresse Byte...	Ändern von Speicherinhalten
Hunt	H Start Ende Byte	Suchen von Bytefolgen
Transfer	T Start Ende Ziel	Verschieben von Speicherbereichen
Copy	C Start Ende Ziel	Vergleichen von Speicherbereichen
Fill	F Start Ende Byte	Füllen von Speicherbereichen
Save	S "name", Gerät, Start, Ende+1	Speichern eines Bereichs
Verify	V "name", Gerät	Vergleichen eines Speicherbereichs

Bild 17. Tabelle der wichtigsten Befehle des TEDMON

III. Dumpen vom Anfang des Basic-Speichers bis zur angegebenen Adresse

```
M 1001 1030
```

Wir sehen uns dann von $1030 rückwärts mit ein wenig Gespür das Memory Dump an und suchen die letzte Endmarkierung. Diese ist das Null-Byte bei $1023. Bei $1026 steht dann die Zeilennummer: »6E 00«.

In Basic rechnen wir $006E mit <PRINT DEC ("006E")> um und erhalten 110.

In Zeile 110 steht demzufolge der gesuchte Text.

Hat der H-Befehl in Schritt II mehrere Adressen genannt, ist Schritt III für jede Adresse durchzuführen.

LISTschutz durch ASCII-Codes von Steuerzeichen

Es gibt nicht nur für die druckenden Zeichen, sondern auch für die Steuerzeichen ASCII-Codes. In Bild 8 können Sie noch einmal nachsehen, wenn Sie Ihr Wissen auffrischen wollen.

Wir werden besonders wirkungsvolle Steuercodes in REM-Zeilen einbauen, da sie dort keinen Schaden anrichten (Texte nach REM werden nicht ausgeführt). Das Prinzip ist klar: Bei der Eingabe einer REM-Zeile werden zunächst Kommentare eingegeben, die später mit Steuercodes überschrieben werden. Beim LISTen der Zeile werden diese Steuercodes dann ausgeführt.

In Bild 15 sehen Sie ein Memory Dump einer manipulierten Form von »Versuchskaninchen«. Bitte geben Sie einmal dieses Memory Dump ein und lassen Sie dann den Basic-Befehl <LIST> ausführen, um alle Effekte zu sehen. Warum dies so ist, können Sie sich mit Hilfe der Steuerzeichentabelle selbst erklären.

Die verwendeten Steuercodes haben nur Werte bis $7F. Codes ab $80 würde der Computer als Token interpretieren und anstelle eines Steuerzeichens einen Basic-Befehl ausgeben. Dies umgeht man, indem man vor einem Steuerzeichen, das größer als $7F ist, folgende Bytes setzt:

»22 14 0D«

Pro REM-Zeile sollte man dies nur einmal machen. Dann sind alle Steuerzeichen erlaubt. Die genaue Funktionsweise dieser drei Byte wollen wir hier nicht ausbreiten, nur soviel: Der Computer wird dazu gebracht, jedes Zeichen, daß dieser Sequenz folgt, auszugeben, ohne es vorher zu decodieren.

Als Bild 16 finden Sie eine Version von Versuchskaninchen, in der dieser Trick angewendet wurde. Außerdem wird von den ESC-Sequenzen (siehe Handbuch) Gebrauch gemacht.

Wir sind nun am Ende des Kurses angelangt. Wir hoffen, es hat Ihnen Spaß gemacht, Ihren Computer an einigen Stellen zu durchschauen und »auszutricksen«.

In Bild 17 sehen Sie noch eine Tabelle mit allen Befehlen des TEDMON. Der C 128 enthält übrigens im Gegensatz zum C 64 auch einen eingebauten Maschinensprache-Monitor. Die Befehle fast aller Monitore sind von der Syntax fast identisch, so daß keine Probleme mit anderen Produkten auftreten dürften!

Sicherlich werden Sie noch Zeit brauchen, bis Sie mit dem Monitor so selbstverständlich wie mit Basic umgehen können, aber dann sind Sie bereits so weit, daß Sie mit Maschinensprache beginnen können.

(F. Müller/ks)



Dateiverwaltung mit dem C 16 in Maschinensprache

Um Ihre Basic-Programme zu beschleunigen, sind unterstützende Maschinenroutinen sehr hilfreich. Am Beispiel »Dateiverwaltung« zeigen wir Ihnen, wie man so etwas macht.

Dieser Artikel behandelt die Datenein- und -ausgabe in Maschinensprache mit Hilfe der im Betriebssystem des C 16 vorhandenen Routinen. Eine dieser Routinen kennt wohl jeder Assemblerprogrammierer: Die Routine BSOUT ($FFD2), die häufig zur Ausgabe von Zeichen auf dem Bildschirm verwendet wird. Eine weitere bekannte Routine ist GETIN ($FFE4), die ein Zeichen von der Tastatur einliest.

Wenn Sie sich in Ihren Assemblerprogrammen auf Tastatur und Bildschirm als Peripheriegeräte beschränken, genügen Ihnen die genannten Routinen. Sollten Sie jedoch beabsichtigen, komplette Anwenderprogramme in Assembler zu schreiben, zum Beispiel eine Dateiverwaltung oder Textverarbeitung, müssen Sie zuvor folgende Probleme lösen:

1. Wie kann ich Daten statt auf den Bildschirm an beliebige Peripheriegeräte senden oder von diesen einlesen?
2. Wie erkenne ich bei Disketten- oder Datasettendateien das Dateiende, das in Basic mit der Statusvariablen ST abgefragt werden kann?
3. Woran erkenne ich, ob während der Datenübertragung Fehler auftraten?

Dieser Artikel wird – hoffentlich – alle derartigen Fragen beantworten. Zuvor will ich jedoch noch klären, warum es vorteilhaft ist, eine Dateiverwaltung in Maschinensprache zu programmieren, selbst wenn Sie vielleicht nie ein komplettes Anwenderprogramm in Maschinensprache schreiben werden.

Die meisten größeren Programme für den C 16 werden Sie wahrscheinlich in Basic schreiben, da es außerordentlich aufwendig – und fehlerbehaftet – ist, ein größeres Programm vollständig in Maschinensprache zu erstellen. Am häufigsten wird Maschinensprache für kleinere Unterprogramme verwendet, um zeitkritische Basic-Programmteile zu ersetzen. Paradebeispiel für dieses Einsatzgebiet sind Sortierroutinen, die – in Basic programmiert – bei großen Datenmengen oftmals recht einschläfernd sein können.

Ein weiteres Einsatzgebiet sind Assemblerroutinen, die Mängel des eingebauten Basic-Interpreters beheben sollen, zum Beispiel eigene INPUT-Routinen.

Viele nützliche Assemblerroutinen können jedoch nur dann erstellt werden, wenn Sie die Datenein- und ausgabe auf der Maschinenspracheebene beherrschen. Vorstellbar wäre zum Beispiel ein sogenannter Drucker-Spooler, ein Assemblerprogramm, das in den Systeminterrupt eingebunden ist, der jede sechzigstel Sekunde erfolgt. Bei jedem Interrupt wird ein Zeichen des auszudruckenden Programmlistings oder Textes an den Drucker geschickt. Während der Text durch diese »im Hintergrund laufende« Interruptroutine ausgedruckt wird, können Sie ungehindert mit Ihrem C 16 arbeiten, zum Beispiel ein Basic-Programm ablaufen lassen oder editieren.

Ein weiteres Beispiel wäre eine INPUT #-Routine, die einen String von Kassette oder Diskette einliest. Wozu, werden Sie sich fragen, da doch das C 16-Basic den Befehl INPUT # bereits besitzt? Nun, jeder der sich bereits intensiver mit Dateiverwaltung beschäftigt hat, weiß, daß INPUT # den Programmierern des Basic-Interpreters gewiß keinen Ruhm einbrachte. Dieser Befehl arbeitet ebenso katastrophal wie der Befehl INPUT: Soll ein String eingelesen werden, der länger ist als 80 Zeichen ist, erscheint die Fehlermeldung »String too long error«. Enthält der String ein Komma oder ein Semikolon, geht das Einlesen schief, da beide Zeichen ebenso wie ein Return (chr$(13)) als Stringendemarke aufgefaßt werden.

Im folgenden Artikel werde ich die wichtigsten Betriebssystemroutinen des C 16/C 116 zur Datenein- beziehungsweise Datenausgabe vorstellen. Bei der Beschreibung dieser Routinen werden die Funktionen erläutert, was bei Verwendung der jeweiligen Routine zu beachten ist (Startadresse; Übergabeparameter), und welche Parameter – die vor allem zur Erkennung eventuell aufgetretener Fehler dienen – nach dem Aufruf der Routinen zurückübergeben werden.

Alle Beispielprogramme können problemlos mit dem Monitor des C 16 eingegeben werden. Sowohl der Aufruf als auch die Startadressen der verwendeten Betriebssystemroutinen sind ohne jede Änderung auch auf den C 64 und den C 128 übertragbar, im Gegensatz zu den im letzten Teil des Artikels erläuterten Routinen des Basic-Interpreters, deren Einsprungadressen – und teilweise auch deren Funktion – leider von Computer zu Computer unterschiedlich ist.

Den Abschluß dieses Kapitels bildet die Programmierung einer INPUT #-Routine, die Sie in all Ihren Programmen als Ersatz für den ziemlich mangelhaften Basic-Befehl INPUT # einsetzen können, und die in der Lage ist, Strings bis zu einer Maximallänge von 255 Zeichen einzulesen, wobei der String im Gegensatz zum INPUT # des Basic-Interpreters beliebige Zeichen enthalten kann.

Öffnen einer Datei

Sie alle kennen den Basic-Befehl OPEN, mit dem eine Datei für Lese- oder Schreibzugriffe geöffnet werden kann. Erst nach dem Öffnen einer Datei kann die eigentliche Datenein- oder -ausgabe mit PRINT # oder INPUT # erfolgen. Die Programmierung auf der Maschinenspracheebene verläuft analog. Zuerst muß eine Datei geöffnet werden, wobei die gleichen Dateiparameter angegeben werden wie im OPEN-Befehl, dessen genaue Syntax bekanntlich lautet:

OPEN (LF), (GA), (SA), "DATEINAME, TYP, MODUS"

LF = Logische Filenummer
GA = Geräteadresse
SA = Sekundäradresse

Dieser für die Dateibehandlung grundlegende Basic-Befehl kann leider nur mit großem Aufwand in die entsprechenden Maschinenbefehle umgesetzt werden. Sollten Sie annehmen, daß hierzu der Aufruf einer einzigen Betriebssystemroutine ausreicht, muß ich Sie leider enttäuschen. Im Betriebssystem existiert zwar eine Routine mit dem Namen OPEN. Bevor diese Routine aufgerufen werden kann, müssen zuvor zwei Vorbereitungsroutinen aufgerufen werden, mit denen die benötigten Dateiparameter übergeben werden.

Mit der zuerst aufzurufenden Routine SETFLS ($FFBA) werden die Parameter logische Filenummer, Geräteadresse und Sekundäradresse übergeben. Anschließend wird die Routine SETNAM ($FFBD) aufgerufen, die für die Übergabe des Dateinamens zuständig ist (inklusive der Zusätze Typ und Modus). Erst nach dieser Vorbereitung kann die OPEN-Routine ($FFC0) aufgerufen werden.

SETFLS ($FFBA): Fileparameter setzen

Der erste Schritt zum Öffnen einer Datei – gleich ob zum Lesen oder Schreiben von Daten – ist die Übergabe der Parameter logische Filenummer, Geräteadresse und Sekundäradresse mit der Routine SETFLS.

Als logische Filenummer kann eine beliebige Zahl zwischen 1 und 255 angegeben werden.

Die Geräteadresse liegt üblicherweise zwischen Null und Acht (0=Tastatur; 1=Datasette; 3=Bildschirm; 4=Drucker; 8=Floppy). In den folgenden Programmbeispielen werde ich immer (!) die Geräteadresse Eins verwenden, da die Datasette wohl das am häufigsten in Verbindung mit dem C 16 verwendete Peripheriegerät ist.

Die Bedeutung der Sekundäradresse ist je nach verwendetem Peripheriegerät völlig unterschiedlich. Soll eine Diskettendatei geöffnet werden, kann eine beliebige Sekundäradresse zwischen 2 und 14 angegeben werden.

Bei Druckern bestimmt die angegebene Sekundäradresse häufig den Druckmodus, ob zum Beispiel im Groß-/Grafik- oder im Klein-/Großmodus gedruckt werden soll. Mit welcher Sekundäradresse welcher Druckmodus angesprochen wird, ist von Drucker zu Drucker unterschiedlich und muß daher im zugehörigen Druckerhandbuch nachgelesen werden.

Beim Öffnen einer Datasettendatei kann eine Sekundäradresse zwischen Null und Zwei angegeben werden. Null bedeutet, daß anschließend Daten vom Band gelesen werden sollen, 1 geben Sie an, wenn Schreibzugriffe erfolgen sollen. Die Angabe einer 2 bedeutet ebenfalls, daß anschließend Schreibzugriffe erfolgen. Zusätzlich wird eine sogenannte »EOT«-Markierung auf das Band geschrieben, eine Markierung, die angibt, daß sich die Datei am Bandende befindet und keine weitere Datei folgt. Sinn und Zweck dieser Markierung blieb mir persönlich jedoch etwas schleierhaft, da ich sie in meinen Programmen noch nie benötigt habe. Auch in den folgenden Beispielprogrammen werde ich daher ausschließlich die Sekundäradressen Null und Eins verwenden.

Sie wissen nun, welche Parameter der Routine SETFLS übergeben werden müssen, nicht jedoch wie. Alle drei Parameter werden mit Hilfe der Prozessorregister übergeben:

Akku = logische Filenummer (1 bis 255)
X-Register = Geräteadresse (0, 1, 3, 4 oder 8)
Y-Register = Sekundäradresse (Datasette: 0 und 1; Floppy: 2 bis 14)

Wenn wir zum Beispiel Daten in eine Datasettendatei schreiben wollen, übergeben wir der Routine SETFLS folgende Parameter:

```
LDA #$01    ;LOGISCHE FILENUMMER 1
LDX #$01    ;GERÄTEADRESSE 1 = DATASETTE
LDY #$01    ;SEKUNDÄRADRESSE 1 = DATEN AUF BAND
             SCHREIBEN
JSR $FFBA   ;SETFLS AUFRUFEN
```

Wenn wir Daten vom Band lesen wollen, genügt es, den Befehl LDY #$01 abzuändern in LDY #$00.

SETNAM ($FFBD): Dateinamen (+Zusätze) übergeben

Der Dateiname darf aus maximal 16 Zeichen bestehen. Unter »Zusatz« sind die Angaben »Typ« und »Modus« zu verstehen. Beide Angaben werden nur bei Diskettendateien benötigt, um zum Beispiel mit dem Befehl

```
OPEN 2,8,2,"TEST,S,W"
```

die sequentielle Datei »test« zum Schreiben zu öffnen.

Typ = S (Sequentielle Datei), R (Relative Datei), P (Programmdatei) oder U (Userdatei).
Modus = R (»Read«=Daten lesen), W (»Write«=Daten schreiben) oder A (»Append«=Daten an bestehende Datei anhängen)

Die Angabe des Dateityps wird bei Datasettendateien nicht benötigt, da die Datasette nur eine einzige Form der Datendatei kennt, den sequentiellen Typ.

Die Angabe der Zugriffsart, des Modus, ist bei Verwendung der Datasette ebenfalls überflüssig, da diese Angabe bereits beim Aufruf von SETFLS erfolgte (Sekundäradresse 0 bis 2). Da wir im folgenden nur mit der Datasette arbeiten werden, können wir uns auf die Angabe des Dateinamens beschränken.

Bei Verwendung der Datasette muß übrigens nicht unbedingt ein Dateiname angegeben werden. Die angelegte Datei erhält dann natürlich keinen Namen. In den folgenden Beispielen werde ich dennoch immer Dateinamen verwenden. Sollten Sie jemals von der Datasette auf eine Floppy umsteigen, bleibt Ihnen dadurch die Umgewöhnung an Dateinamen erspart.

Die Parameter für SETNAM werden teilweise ebenfalls mit Hilfe der Prozessorregister übergeben. Zusätzlich muß jedoch in einem beliebigen Speicherbereich, im Programm selbst, in der Zeropage, oder an einem beliebigen anderen Ort der Dateiname in ASCII-Form abgelegt sein. Wenn Sie die RS232-Schnittstelle nicht benutzen (um zum Beispiel ein Modem zu betreiben), bietet sich der Bereich $03F7 bis $0436 (dezimal: 1015 bis 1078) an, der ausschließlich von dieser Schnittstelle verwendet wird. Wenn der Dateiname in diesem Bereich abgelegt wurde, wird der Akku mit der Länge des Namens geladen. Das X- beziehungsweise Y-Register enthält einen Zeiger auf die Adresse, an der sich der Name befindet (X-Register=Low-Byte der Adresse; Y-Register=High-Byte):

```
LDA #$04    ;LÄNGE DES NAMENS: 4 ZEICHEN
LDX #$F7    ;LOW-BYTE DER ADRESSE $03F7
LDY #$03    ;HIGH-BYTE DER ADRESSE $03F7
JSR $FFBD   ;SETNAM AUFRUFEN
```

Diese Routine ist nur dann lauffähig, wenn zuvor ein Dateiname mit einer Länge von vier Zeichen in ASCII-Form ab $03F7 abgelegt wurde.

OPEN ($FFC0): Datei öffnen

Nach diesen langwierigen Vorbereitungen kann endlich die OPEN-Routine des Betriebssystems aufgerufen werden. Der Aufruf mit JSR $FFC0 genügt, da ausnahmsweise keine weiteren Parameter übergeben werden müssen. Die logische Datei wird nun geöffnet. Dabei ist es jedoch möglich, daß ein Fehler auftritt, zum Beispiel wenn auf Band geschrieben werden soll und die Datasette noch im Schrank liegt. In diesem Fall tritt ein »device not present error« auf.

Wenn Sie wirklich professionell programmieren wollen, dürfen Sie eventuell auftretende Fehler selbstverständlich nicht einfach ignorieren, sondern benötigen eine sogenannte »Fehlerbehandlungsroutine«, zu der Ihr Programm in einem solchen Fall verzweigt (zum Beispiel, um den Benutzer aufzufordern, endlich die Floppy einzuschalten oder eine Diskette einzulegen). Übrigens: Da ich nicht von mir behaupte, professionell zu programmieren, habe ich es mir erlaubt, in der noch vorzustellenden INPUT#-Routine auf eine solche Fehlerbehandlung zu verzichten. Sollten Sie weniger bequem veranlagt sein, dürfen Sie das Programm jedoch gern entsprechend erweitern (und mir natürlich die erweiterte Version zuschicken).

Ob beim Öffnen einer Datei ein Fehler autrat, erkennen Sie am Carry-Flag. Bei fehlerfreiem Öffnen der Datei ist es nach der Rückkehr aus der OPEN-Routine gelöscht. Trat ein Fehler auf, ist das Carry-Flag gesetzt und im Akku wird die Nummer

des Fehlers übergeben. Bei gesetztem Carry-Flag sollte Ihr Programm daher zu einer Fehlerbehandlungsroutine verzweigen:

```
JSR $FFC0 ;OPEN AUFRUFEN
BCS $???? ;ADRESSE DER FEHLERBEHANDLUNGSROUTINE
```

Die Fehlernummern haben folgende Bedeutung:

0 = STOP-Taste gedrückt
1 = too many files
2 = file open
3 = file not open
4 = file not found
5 = device not present
6 = not input file
7 = not output file
8 = missing filename
9 = illegal device number

Die Fehler 3, 6 und 7 sind beim Öffnen einer Datei bedeutungslos. Sie können nur nach Lese- oder Schreibversuchen auftreten. Ein Fehler wie zum Beispiel »missing filename« ist dagegen gerätespezifisch. Bei Verwendung der Datasette müssen Sie nicht unbedingt einen Dateinamen angeben, daher kann auch dieser Fehler nicht auftreten.

CHKIN ($FFC6) und CKOUT ($FFC9): Ein-/Ausgaben umleiten

Ich habe behauptet, daß das Maschinensprache-äquivalent zum Basic-Befehl OPEN die Routinen SETFLS, SETNAM und der anschließende Aufruf der OPEN-Routine sei. Leider müssen wir noch zwei weitere Routinen besprechen, bevor wir zur eigentlichen Datenein- beziehungsweise -ausgabe kommen, die Routinen CHKIN und CKOUT.

Wie Sie wissen, ist die Tastatur beim C 16 das Standardeingabegerät und der Bildschirm das Standardausgabegerät. Um nach dem Öffnen einer Datei in diese Daten zu schreiben oder Daten daraus zu lesen, muß die Ein- beziehungsweise Ausgabe auf die geöffnete logische Datei umgelenkt werden, da sonst die Standardgeräte angesprochen werden.

Das Umlenken der Eingabe von der Tastatur auf die mit der OPEN-Routine geöffnete Datei erfolgt mit der Routine CHKIN, der im X-Register die beim Aufruf von SETFLS verwendete logische Filenummer übergeben wird, zum Beispiel die 1:

```
LDX #$01      ;LOGISCHE FILENUMMER
JSR $FFC6     ;AUFRUF VON CHKIN
```

Um in eine geöffnete Datei Daten zu schreiben, muß die Ausgabe vom Standardgerät Bildschirm zuvor auf die geöffnete logische Datei umgeleitet werden. Der zuständigen Routine CKOUT wird ebenfalls im X-Register die verwendete logische Filenummer übergeben:

```
LDX #$01      ;LOGISCHE FILENUMMER
JSR $FFC9     ;AUFRUF VON CKOUT
```

Beide Routinen zeigen auftretende Fehler ebenso wie die OPEN-Routine durch ein gesetztes Carry-Flag an. Trat ein Fehler auf, wird die Fehlernummer ebenfalls im Akku übergeben.

Demoprogramm 1: Datei in Maschinensprache öffnen

Wir sind nun endlich soweit, das theoretisch erworbene Wissen in ein Programm umzusetzen. Mit den besprochenen Routinen können wir ein Demonstrationsprogramm schreiben, das in Maschinensprache eine logische Datei zum Schreiben von Daten auf Band öffnet. Als Dateinamen verwenden wir »test«, als logische Filenummer eine Eins. Da eine Datasettendatei zum Schreiben geöffnet werden soll, müssen wir die Geräteadresse Eins (=Datasette) und die Sekundäradresse Eins (=Schreiben) verwenden.

Zum Schreiben der Daten verwenden wir ein kleines Basic-Programm, das zusätzlich die Aufgabe übernehmen soll, den Dateinamen »test« in ASCII-Form in den RS232-Puffer zu POKEn, bevor es unsere »Dateiöffnungsroutine« aufruft und anschließend Daten in die – durch die Maschinenroutine geöffnete – Datei schreibt:

```
100 REM *** DATEN SCHREIBEN ***
110 DN$="TEST":REM DATEINAME
120 FOR I=1 TO LEN(DN$)
130 : POKE 1014+I,ASC(MID$(DN$,I,1))
140 NEXT
150 SYS 1630:REM MASCHINENROUT.AUFRUFEN
160 PRINT#1,"DIES IST"
170 PRINT#1,"EIN TEST"
180 CLOSE 1
190 :
200 :
210 REM *** DATEN LESEN ***
220 PRINT"BAND BITTE ZURUECKSPULEN"
230 PRINT"DANACH 'RETURN' DRUECKEN"
240 GET A$:IF A$<>CHR$(13) THEN 240
250 SCNCLR:
260 :
270 OPEN 1,1,0,"TEST":REM DATEI OEFFNEN
280 INPUT#1,A$:PRINT A$
290 INPUT#1,B$:PRINT B$
300 CLOSE 1
```

Die Maschinenroutine geben Sie bitte mit dem im C 16 integrierten Monitor ab $065E (dezimal 1630) ein:

```
. 065e  LDA #$01    ;LOGISCHE FILENUMMER 1
. 0660  LDX #$01    ;GERAETEADRESSE 1 = DATASETTE
. 0662  LDY #$01    ;SEKUNDAERADRESSE 1 =
                    DATEN SCHREIBEN
. 0664  JSR $FFBA   ;SETFLS AUFRUFEN
. 0667  LDA #$04    ;LAENGE DES NAMENS: 4 ZEICHEN
. 0669  LDX #$F7    ;LOW-BYTE DER ADRESSE $03F7
. 066b  LDY #$03    ;HIGH-BYTE DER ADRESSE $03f7
. 066d  JSR $FFBD   ;SETNAM AUFRUFEN
. 0670  JSR $FFC0   ;OPEN AUFRUFEN
```

Tabelle der verwendeten Betriebssystem- und Interpreterroutinen

Name	Funktion	Parameter hin	Parameter zurück	Adresse
SETFLS	Fileparameter setzen	AKKU=LF; X=GA; Y=SA		$FFBA
SETNAM	Dateiname übertragen	AKKU=Länge; X/Y=Point. auf Name		$FFBD
OPEN	Log. Datei öffnen	Vorbereitung: SETFLS, SETNAM	Fehler: SEC, Fehlernr. im AKKU	$FFC0
CHKIN	Eingabe von Log. Datei	X=Log. Filenummer	Fehler: SEC, Fehlernr. im AKKU	$FFC6
CKOUT	Ausgabe auf Log. Datei	X=Log. Filenummer	Fehler: SEC, Fehlernr. im AKKU	$FFC9
READST	I/O-Status abfragen		Status im AKKU (BIT 6 = Dateiende)	$FFB7
BASIN	Zeichen einlesen		Zeichen im AKKU	$FFCF
BSOUT	Zeichen ausgeben	Zeichen im AKKU		$FFD2
CHKKOM	Komma lesen			$9491
GETBYT	Ein-Byte-Wert lesen		X=übergebenes Byte	$9D84
STRPOS	Variablenadresse holen		$47/$48=Point. auf Längendescr.	$96A5
STRRES	Stringplatz reservieren	AKKU=Stringlänge		$A906

```
. 0673  LDX #$01     ;LOGISCHE FILENUMMER
. 0675  JSR $FFC9    ;AUFRUF VON CKOUT
. 0678  RTS          ;ZURUECK ZUM BASIC
```

Das Maschinenprogramm besteht im Grunde nur aus einer Zusammenfassung der bereits beschriebenen Teile, das heißt aus dem Aufruf von SETFLS, SETNAM, OPEN und CKOUT. Beachten Sie bitte, daß eventuell auftretende Fehler ignoriert werden (Carry-Flag nach OPEN und CKOUT abfragen).

Geben Sie dieses Programm bitte mit dem Monitor ein (lassen Sie die Kommentare jedoch weg und versuchen Sie nicht, die einzelnen Programmteile wie abgebildet durch Leerzeilen optisch zu unterteilen!), geben Sie anschließend das Basic-Programm ein und starten Sie es mit
RUN

Zum Programmablauf: In den Zeilen 110 bis 140 werden mit einer Schleife die ASCII-Codes der einzelnen Zeichen des Dateinamens »test« nach 1015 bis 1018 gePOKEt. Anschließend wird unsere Maschinenroutine aufgerufen, die eine Datasettendatei für Schreibzugriffe mit der logischen Filenummer 1 öffnet (Zeile 150). Unter Angabe dieser Filenummer schreibt das Basic-Programm nun zwei Strings in die Datei, bevor sie wieder geschlossen wird (Zeilen 160 bis 180).

Vor dem Wiedereinlesen der Daten fordert Sie das Basic-Programm auf, zum Dateianfang zurückzuspulen und anschließend
RETURN
zu drücken. Nach RETURN wird die Warteschleife in Zeile 240 verlassen und der Bildschirm gelöscht, bevor die Datei »test« zum Lesen geöffnet wird (Zeilen 240 bis 270). Die beiden Strings werden eingelesen, auf dem Bildschirm ausgegeben und die Datei geschlossen.

Wie Sie sehen, ist es zweifellos bequemer, einen Basic-Befehl zum Öffnen einer Datei zu verwenden, als ein Maschinenprogramm, das an mehrere Routinen die verschiedensten Parameter übergeben muß, um die gleiche Funktion zu erfüllen. Die im Laufe dieses Kapitels vorgestellte INPUT#-Routine entschädigt Sie jedoch für diesen Aufwand. In Basic könnte diese Routine nur mit einem kleinen Unterprogramm unter Verwendung von GET# erstellt werden und wäre weitaus langsamer als die entsprechende Maschinenroutine. (Versuchen Sie einmal, 20 Strings mit einer Länge von jeweils 100 Zeichen mit GET# nach und nach einzulesen. Sie werden staunen, wie langsam Ihre Datasette plötzlich wird.)

BSOUT ($FFD2): Datenausgabe auf logische Datei

Wir können nun zwar in Maschinensprache beliebige Dateien öffnen, bisher jedoch weder Daten ausgeben noch einlesen. Sie alle wissen sicherlich, daß mit der Routine BSOUT Daten auf dem Bildschirm ausgegeben werden können. BSOUT gibt ein im Akku übergebenes Zeichen (ASCII-Code!) auf dem Bildschirm aus, und zwar an der aktuellen Cursorposition.

BSOUT ist jedoch keineswegs eine speziell zur Ausgabe auf dem Bildschirm gedachte Routine, sondern erfüllt die weitaus allgemeinere Funktion, Daten in eine beliebige logische Datei zu schreiben. Diese Datei ist normalerweise der Bildschirm, wir können mit BSOUT jedoch auch Zeichen in eine Datasetten-, Disketten- oder Druckerdatei schreiben.

Die benötigten Vorbereitungsroutinen kennen wir bereits. Zuerst wird eine logische Datei geöffnet, anschließend die Datenausgabe mit CKOUT auf diese Datei umgeleitet. Wenn nun BSOUT aufgerufen wird, werden die im Akku übergebenen Zeichen dank der Umleitung nicht auf dem Bildschirm ausgegeben, sondern in die angegebene Datei geschrieben.

```
...          ;DATASETTENDATEI ZUM
...          ;SCHREIBEN OEFFNEN
...          ;EINGABE AUF DIE DATEI LEGEN
LDA #$41     ;ASCII-CODE VON »A«
JSR $FFD2    ;»A« IN DIE DATEI SCHREIBEN
...
...
...
```

BASIN ($FFCF): Dateneingabe von logischer Datei

BASIN erfüllt die entgegengesetzte Funktion, das Einlesen von Daten aus einer beliebigen logischen Datei. Wenn die Zeichen nicht vom Standardgerät Tastatur gelesen werden sollen, muß ebenfalls eine logische Datei zum Lesen geöffnet und die Eingabe mit CHKIN auf diese Datei umgelenkt werden. Nun kann mit BASIN Zeichen für Zeichen aus dieser Datei gelesen werden. Die eingelesenen Zeichen werden im Akku übergeben.

```
...          ;DATASETTENDATEI ZUM
...          ;LESEN OEFFNEN
...          ;EINGABE AUF DIE DATEI LEGEN
JSR $FFCF    ;EIN ZEICHEN AUS DER DATEI LESEN
STA $xxx1    ;ZEICHEN SPEICHERN
JSR $FFCF    ;NAECHSTES ZEICHEN LESEN
STA $xxx2    ;EBENFALLS SPEICHERN
...
...
```

CLRCH ($FFCC): Standardein-/-ausgabegeräte setzen

Nach beendeter Ein- beziehungsweise Ausgabe von Daten sollte die Routine CLRCH aufgerufen werden, die die Eingabe wieder auf das Standardgerät Tastatur und die Ausgabe auf den Bildschirm umleitet. Es werden keinerlei Vorbereitungsroutinen oder Übergabeparameter benötigt.

```
...          ;LOGISCHE DATEI OEFFNEN
...          ;EIN- ODER AUSGABE UMLENKEN
...          ;DATEN LESEN ODER SCHREIBEN
JSR $FFCC    ;STANDARDEIN-/-AUSGABEGERAETE SETZEN
...          ;(TASTATUR/BILDSCHIRM)
...
...
```

CLOSE ($FFC3): Logische Datei schließen

CLRCH leitet weitere Ein- oder Ausgaben zwar wieder auf die Tastatur beziehungsweise den Bildschirm um, die zuvor geöffnete logische Datei wird jedoch nicht automatisch geschlossen! Zum Abschluß jeder Schreib- oder Leseroutine sollte daher die Routine CLOSE aufgerufen werden, die die angegebene logische Datei schließt, wobei die Filenummer im Akku übergeben werden muß.

```
...          ;DATEI MIT LOGISCHER FILENUMMER 1 OEFFNEN
...          ;EIN-/AUSGABE UMLEITEN
...          ;DATEN LESEN ODER SCHREIBEN
JSR $FFCC    ;EIN-/AUSGABE AUF STANDARDGERAETE
LDA #$01     ;LOGISCHE FILENUMMER
JSR $FFC3    ;LOGISCHE DATEI SCHLIESSEN
...
...
...
```

READST ($FFB7): Status abfragen

Ein Problem müssen wir noch lösen, bevor wir ausschließlich in Maschinensprache Dateiverwaltung betreiben können, und zwar die Abfrage des Dateiendes. Das Einlesen einer Datei muß beendet werden, wenn das Dateiende erreicht wurde. In Basic können wir dazu die Statusvariable ST abfragen, die den Wert 64 annimmt, wenn das Dateiende erreicht wurde.

In Maschinensprache verwenden wir die Routine READST, die die gleiche Funktion erfüllt (der Basic-Interpreter verwendet diese Routine selbst, um der Variablen ST den jeweiligen Status mitzuteilen).

Nach dem Aufruf von READST wird der aktuelle Status im Akku übergeben. Bei Erreichen des Dateiendes wird Bit 6 gesetzt (da dieses Bit den dezimalen Wert 64 besitzt, wird

die Analogie zur Basic-Variablen ST deutlich).

```
...
...
...
LABEL JSR $FFCF ;DATEN LESEN
      JSR $FFB7 ;READST AUFRUFEN
      AND #$40  ;ALLE BITS AUSSER BIT 6
                 AUSMASKIEREN
      BEQ LABEL ;WEITERLESEN, WENN NICHT DATEIENDE
...
...
...
```

```
. 065e  a9 ff     lda #$ff
. 0660  20 06 a9  jsr $a906
. 0663  20 91 94  jsr $9491
. 0666  20 84 9d  jsr $9d84
. 0669  20 c6 ff  jsr $ffc6
. 066c  a0 00     ldy #$00
. 066e  20 cf ff  jsr $ffcf
. 0671  c9 0d     cmp #$0d
. 0673  f0 05     beq $067a
. 0675  91 33     sta ($33),y
. 0677  c8        iny
. 0678  d0 f4     bne $066e
. 067a  84 d0     sty $d0
. 067c  20 cc ff  jsr $ffcc
. 067f  a5 34     lda $34
. 0681  85 d2     sta $d2
. 0683  a9 ff     lda #$ff
. 0685  38        sec
. 0686  e5 d0     sbc $d0
. 0688  18        clc
. 0689  65 33     adc $33
. 068b  85 d1     sta $d1
. 068d  90 02     bcc $0691
. 068f  e6 d2     inc $d2
. 0691  a4 d0     ldy $d0
. 0693  88        dey
. 0694  b1 33     lda ($33),y
. 0696  91 d1     sta ($d1),y
. 0698  88        dey
. 0699  c0 ff     cpy #$ff
. 069b  d0 f7     bne $0694
. 069d  20 91 94  jsr $9491
. 06a0  20 a5 96  jsr $96a5
. 06a3  a0 02     ldy #$02
. 06a5  b1 47     lda ($47),y
. 06a7  99 d3 00  sta $00d3,y
. 06aa  88        dey
. 06ab  10 f8     bpl $06a5
. 06ad  a5 d3     lda $d3
. 06af  f0 08     beq $06b9
. 06b1  a8        tay
. 06b2  91 d4     sta ($d4),y
. 06b4  c8        iny
. 06b5  a9 ff     lda #$ff
. 06b7  91 d4     sta ($d4),y
. 06b9  a4 d0     ldy $d0
. 06bb  a5 47     lda $47
. 06bd  91 d1     sta ($d1),y
. 06bf  c8        iny
. 06c0  a5 48     lda $48
. 06c2  91 d1     sta ($d1),y
. 06c4  a0 00     ldy #$00
. 06c6  a5 d0     lda $d0
. 06c8  91 47     sta ($47),y
. 06ca  c8        iny
. 06cb  a5 d1     lda $d1
. 06cd  91 47     sta ($47),y
. 06cf  85 33     sta $33
. 06d1  c8        iny
. 06d2  a5 d2     lda $d2
. 06d4  91 47     sta ($47),y
. 06d6  85 34     sta $34
. 06d8  60        rts
```

Listing 1. »Input-Routine« in Maschinensprache. Bitte mit dem eingebauten Monitor eingeben.

Demoprogramm 2: Daten in Maschinensprache schreiben und lesen

Die Zeiten, in denen wir uns zum Schreiben und Lesen von Daten mit Basic behelfen mußten, sind nun endgültig vorbei. In diesem zweiten Programmbeispiel werden wir die Strings »DIES IST« und »EIN TEST« in Maschinensprache auf Band schreiben und wieder lesen.

Aus reiner Bequemlichkeit wird zusätzlich ein kleines Basic-Programm verwendet, das den Dateinamen »TEST« ab 1015 und die zu schreibenden Strings ab 1019 in den Speicher POKEt, die Routine zum Schreiben der Strings aufruft, danach die Meldung »ZURUECKSPULEN...« ausgibt, auf die Taste RETURN wartet und zuletzt die Leseroutine aufruft, die beide Strings aus der Datei einliest und auf dem Bildschirm ausgibt.

```
100 REM *** STRINGS VORBEREITEN ***
110 FOR I=1 TO 4:POKE 1014+I,ASC(MID$("TEST",I,1))
    :NEXT
120 A$="DIES IST"+CHR$(13)+"EIN TEST"+CHR$(13)
    +CHR$(0)
130 FOR I=1 TO LEN(A$):POKE 1018+I,ASC
    (MID$(A$,I,1)):NEXT
140 :
150 REM *** STRINGS SCHREIBEN/LESEN ***
160 SYS 1630:REM STRINGS SCHREIBEN
170 PRINT"ZURUECKSPULEN + 'RETURN' DRUECKEN"
180 GET A$:IF A$<>CHR$(13) THEN 180
190 SYS 1678:REM STRINGS LESEN/AUSGEBEN
```

Beachten Sie bitte, daß bei der zeichenweisen Ausgabe mit BSOUT im Gegensatz zur Basic-Ausgabe mit PRINT# nicht automatisch nach jedem String das Zeichen CHR$(13) (Carriage Return oder auch Zeilenvorschub) auf das Band geschrieben wird. Die Basic-Routine fügt dieses Zeichen daher an das Ende beider Zeichenketten an (Zeile 120). Als letztes Zeichen wird ein CHR$(0) als Endemarke angehängt, an dem unsere Maschinenroutine das Ende des auszugebenden Strings erkennen soll.

Teil 1: Daten schreiben

```
. 065e LDA #$01      ;LOGISCHE FILENUMMER
. 0660 LDX #$01      ;GERAETEADRESSE
. 0662 LDY #$01      ;SEKUNDAERADRESSE (1=WRITE)
. 0664 JSR $ffBA     ;SETLFS AUFRUFEN

. 0667 LDA #$04      ;LAENGE DES DATEINAMENS
. 0669 LDX #$f7      ;LOW-BYTE ADRESSE DATEINAME
. 066b LDY #$03      ;HIGH-BYTE ADRESSE DATEINAME
. 066d JSR $FFBD     ;SETNAM AUFRUFEN

. 0670 JSR $FFC0     ;OPEN AUFRUFEN
. 0673 LDX #$01      ;LOGISCHE FILENUMMER
. 0675 JSR $FFC9     ;CKOUT AUFRUFEN

. 0678 LDX #$00      ;ZEIGER INITIALISIEREN
. 067A LDA $03FB,X   ;ERSTES ZEICHEN HOLEN
. 067D BEQ $0685     ;FERTIG, WENN ENDEMARKE
. 067f JSR $FFD2     ;ZEICHEN IN DATEI SCHREIBEN
. 0682 INX           ;ZEIGER AUF NAECHSTES ZEICHEN
. 0683 BNE $067A     ;ZUM SCHLEIFENANFANG (IMMER!)

. 0685 JSR $FFCC     ;STANDARDGERAETE SETZEN
. 0688 LDA #$01      ;LOGISCHE FILENUMMER
. 068a JSR $FFC3     ;LOGISCHE DATEI SCHLIESSEN
. 068d RTS           ;RUECKKEHR ZUM BASIC
```

Teil 2: Daten lesen/ausgeben

```
. 068e LDA #$01      ;LOGISCHE FILENUMMER
. 0690 LDX #$01      ;GERAETEADRESSE
```

```
. 0692 LDY #$00    ;SEKUNDAERADRESSE (0=READ)
. 0694 JSR $FFBA   ;SETLFS AUFRUFEN

. 0697 LDA #$04    ;LAENGE DES DATEINAMENS
. 0699 LDX #$F7    ;LOW-BYTE ADRESSE DATEINAME
. 069b LDY #$03    ;HIGH-BYTE ADRESSE DATEINAME
. 069D JSR $FFBD   ;SETNAM AUFRUFEN

. 06A0 JSR $FFC0   ;OPEN AUFRUFEN
. 06A3 LDX #$01    ;LOGISCHE FILENUMMER
. 06A5 JSR $FFC6   ;CHKIN AUFRUFEN

. 06A8 JSR $FFCF   ;BASIN AUFRUFEN
. 06AB JSR $FFD2   ;ZEICHEN AUSGEBEN
. 06AE JSR $FFB7   ;READST AUFRUFEN
. 06B1 AND #  $40  ;ALLE AUSSER BIT 6 AUSMASKIEREN
. 06B3 BEQ $06A8   ;WEITER, WENN NICHT DATEIENDE

. 06B5 JSR $FFCC   ;STANDARDGERAETE SETZEN
. 06B8 LDA #  $01  ;LOGISCHE FILENUMMER
. 06BA JSR $FFC3   ;LOGISCHE DATEI SCHLIESSEN
. 06BD RTS         ;RUECKKEHR ZUM BASIC
```

Dieses Programm dürfte gut verständlich sein, da es unter Verzicht auf Programmkürze in aller Ausführlichkeit programmiert wurde. Mit dem Öffnen und Schließen der Dateien im ersten und zweiten Programmteil sollten Sie nun vertraut sein.

Wichtig an diesem Programm ist, daß im ersten Teil Zeichen für Zeichen in die Datei geschrieben werden, bis das nächste Zeichen eine Null ist, jene Endemarke, die das Basic-Programm am Ende des Strings in den Speicher POKEte.

Beim Lesen der Zeichen könnte ebenfalls eine solche Endemarke benutzt werden, um das Dateiende festzustellen. Statt dessen wird die erläuterte Routine READST verwendet. Zeichen für Zeichen werden aus der Datei gelesen und auf dem Bildschirm ausgegeben, wobei nach jedem Lesevorgang READST aufgerufen und das sechste Bit des im Akku übergebenen Status-Bytes überprüft wird (AND #$40). Ist dieses Bit gesetzt, wurde das Dateiende erreicht und das Programm kehrt zum Basic zurück (zuvor werden noch CLRCH und CLOSE aufgerufen).

Die Interpreterroutinen CHKKOM, GETBYT, STRPOS und STRRES

Wie ich es Ihnen versprochen habe, will ich im letzten Teil dieses Artikels unsere frisch erworbenen Kentnisse dazu benutzen, eine verbesserte Version der INPUT#-Routine vorzustellen.

Vorher muß ich jedoch kurz auf verschiedene Routinen des Basic-Interpreters eingehen, die für unsere Routine unbedingt benötigt werden. Da der Aufruf dem normalen INPUT#-Befehl entsprechen soll (INPUT#(logische Filenummer), (Übergabestring)), muß unser Programm in der Lage sein, die Filenummer aus dem Basic-Text zu lesen und auf den angegebenen String zuzugreifen, in dem die eingelesenen Daten gespeichert werden sollen.

Unsere Routine wird im RAM-Bereich $065E bis $06EB (dezimal 1630 bis 1771) liegen und wie folgt aufgerufen:

```
SYS 1630,(LOGISCHE FILENUMMER),(ÜBERGABESTRING)
```

zum Beispiel:

```
SYS 1630,1,a$
```

oder

```
SYS 1630,2,X$(7)
```

Da die einzelnen Parameter durch Kommata getrennt sind, benötigen wir zuerst eine Routine, die in der Lage ist, Kommatas aus dem Basic-Programm einzulesen. Diese Routine wird CHKKOM ($9491) genannt. Der Basic-Interpreter führt ständig einen Zeiger auf die momentan bearbeitete Textstelle mit sich. Nach dem SYS-Aufruf weist dieser Zeiger auf das erste Zeichen hinter der angegebenen Adresse, das heißt auf das Komma.

```
10 DATA A9,FF,20,06,A9,20,91,94,20,84
20 DATA 9D,20,C6,FF,A0,00,20,CF,FF,C9
30 DATA 0D,F0,05,91,33,C8,D0,F4,84,D0
40 DATA 20,CC,FF,A5,34,85,D2,A9,FF,38
50 DATA E5,D0,18,65,33,85,D1,90,02,E6
60 DATA D2,A4,D0,88,B1,33,91,D1,88,C0
70 DATA FF,D0,F7,20,91,94,20,A5,96,A0
80 DATA 02,B1,47,99,D3,00,88,10,F8,A5
90 DATA D3,F0,08,A8,91,D4,C8,A9,FF,91
100 DATA D4,A4,D0,A5,47,91,D1,C8,A5,48
110 DATA 91,D1,A0,00,A5,D0,91,47,C8,A5
120 DATA D1,91,47,85,33,C8,A5,D2,91,47
130 DATA 85,34,60
140 FORI=1630TO1752
150 READA$
160 A=DEC(A$)
170 S=S+A
180 POKEI,A
190 NEXT
200 IFS<>17787THENPRINT"FEHLER!":END
210 PRINT"OK."

READY.
```

Listing 2. DATA-Lader zur »Input-Routine«

Der Aufruf von CHKKOM liest dieses Zeichen ein, setzt den »Textpointer« auf das nächste Zeichen im Basic-Programm und überprüft, ob tatsächlich ein Komma gelesen wurde. Wenn ja, endet CHKKOM mit einem RTS-Befehl, sonst wird ein »Syntax error in ...« ausgegeben.

Wenn mit CHKKOM das erste Trennzeichen eingelesen wurde, muß anschließend die angegebene Filenummer von unserem Programm gelesen werden. Die Routine GETBYT ($9D84) liest einen beliebigen Ein-Byte-Wert aus dem Basic-Text und übergibt ihn im X-Register. Der Wert muß übrigens nicht direkt als Zahl angegeben werden, da GETBYT auch Variable verarbeitet, so daß zum Beispiel folgender Aufruf möglich ist:

```
LF=1:SYS 1630,LF,A$
```

Bevor wir an die eigentliche Programmerstellung gehen können, muß ein weiteres Problem gelöst werden: Wo speichern wir die eingelesenen Zeichen? In welcher Form sollen die eingelesenen Daten im Speicher abgelegt werden? Am komfortabelsten ist die Benutzung unserer Routine, wenn sie die eingelesenen Zeichen als String anlegt, auf den das aufrufende Basic-Programm direkt zugreifen kann. Der Name des Übergabestrings sollte ebenso wie beim INPUT#-Befehl beim Aufruf angegeben werden können.

Ein Standardproblem bei der Verbindung von Basic und Maschinenroutinen ist der Zugriff auf Basic-Strings von Maschinensprache aus, das Lesen oder gar Anlegen von Strings. Beides ermöglicht uns die Routine STRPOS ($96A5). Diese Routine liest den beim Aufruf angegebenen Stringnamen aus dem Basic-Text und übergibt in $47 und $48 einen Pointer auf die sogenannten »Stringdescriptoren«.

Wie Sie vielleicht wissen, werden Strings vom Ende des verfügbaren Speichers aus abwärts angelegt und befinden sich im Gegensatz zu numerischen Variablen nicht direkt in der sogenannten »Variablentabelle«, die im Speicher immer unmittelbar dem Basic-Programm folgt. In der Variablentabelle befinden sich jedoch alle Daten, die zum Zugriff auf den eigentlichen String notwendig sind, der Stringname, die Stringlänge und die Adresse, an der sich der String selbst

befindet (Name: 2 Byte; Länge: 1 Byte; Adresse: 2 Byte in der Form Low/High).

STRPOS übergibt einen Zeiger auf diese Descriptoren des angegebenen Strings (und zwar nicht auf das erste Descriptoren-Byte (das erste Namens-Byte), sondern auf das Längen-Byte) und legt ihn an, wenn er bisher noch nicht existiert. Wenn der Aufruf zum Beispiel lautet:

```
SYS 1630,A$(7)
```

und unsere Routine zuerst CHKKOM und anschließend STRPOS aufruft, erhalten wir in $47 und $48 einen Pointer, der auf den Längendescriptor des Strings A$(7) zeigt.

Die letzte benötigte Routine heißt STRRES ($A906). Diese Routine reserviert Speicherplatz für den anzulegenden String. Ein Beispiel: Wir wollen einen String mit einer Länge von 10 Zeichen anlegen. Zuerst muß der Akku mit der Stringlänge geladen werden, anschließend wird STRRES aufgerufen:

```
LDA #$0A    ;STRINGLAENGE: 10 ZEICHEN
JSR $A906   ;AUFRUF VON STRRES
```

In $33 und $34 befindet sich ein Pointer auf den Anfang des sogenannten »Stringstacks«, das heißt auf das erste Zeichen des zuletzt angelegten Strings. Nach dem Aufruf von STRRES mit der Stringlänge Zehn im Akku wird dieser Pointer um den Wert Zehn (+2, wie wir noch sehen werden!) erniedrigt. Der dazukommende String kann nun vor (!) dem zuletzt angelegten gespeichert werden (denken Sie bitte daran, daß die Strings vom Speicherende ausgehend abwärts (!) angelegt werden).

Bevor STRRES den Pointer auf den Anfang der Strings um die übergebene Stringlänge (+2) vermindert, wird jedoch überprüft, ob überhaupt ausreichend Platz vorhanden ist, oder ob der nach unten wachsende Stringstack mit dem Basic-Programm kollidieren und dieses überschreiben würde. Wenn nicht ausreichend Platz vorhanden ist, wird eine Garbage Collection durchgeführt, das heißt nicht mehr benötigter Stringmüll beseitigt. Ist auch anschließend noch nicht ausreichend Platz für den anzulegenden String vorhanden, gibt STRRES die Fehlermeldung »out of memory error« aus.

Endlich: Die INPUT #-Routine

Nun können wir uns endlich an die eigentliche Programmierung wagen. Im Gegensatz zu den bisher verwendeten Monitorlistings werde ich die Entwicklung der INPUT #-Routine anhand von Assemblerlistings demonstrieren, da die Routine doch etwas komplexer als die bisherigen Programmbeispiele ist und Assemblerlistings einfacher zu durchschauen sind. Da Sie jedoch kaum über einen Assembler für den C 16 verfügen, finden Sie am Ende dieses Artikels wieder das gewohnte Monitorlisting, das Sie problemlos mit dem integrierten Monitor eingeben können.

```
;*************************
;* INPUT# FUER C16/C116 *
;* (C) SAID BALOUI, 1986 *
;*************************
        .BA $065E    ;PROGRAMMSTART
        .OS          ;OBJECTCODE GENERIEREN
;
LENGTH  .DE $D0      ;STRINGLAENGE
POINTR  .DE $D1      ;HILFSPOINTER
DESCR   .DE $D3      ;STRINGDESCRIPTOREN
DESPOI  .DE $47      ;POINTER AUF DESCRIPTOREN
CHKKOM  .DE $9491    ;KOMMA EINLESEN
GETBYT  .DE $9D84    ;BYTEWERT LESEN
CHKIN   .DE $FFC6    ;INPUT AUF LOGISCHE DATEI
CLRCH   .DE $FFCC    ;STANDARDGERAETE SETZEN
BASIN   .DE $FFCF    ;ZEICHEN AUS LOG.DATEI LESEN
STREND  .DE $33      ;POINTER AUF STRINGSTACK
STRPOS  .DE $96A5    ;STRINGVARIABLE LESEN
STRRES  .DE $A906    ;STRINGPLATZ RESERVIEREN
;
;
*** PLATZ RESERVIEREN ***
     LDA #255      ;255 BYTE
     JSR STRRES    ;RESERVIEREN
```

Diese beiden Befehle reservieren Platz für den einzulesenden String mit einer Maximallänge von 255 Zeichen. Dank der Verwendung von STRRES wird sich das Programm keinesfalls »aufhängen«. Sollte das Basic-Programm, in dem Sie die INPUT #-Routine verwenden, zu umfangreich und daher nicht ausreichend Platz vorhanden sein, wird es sich mit »out of memory error« verabschieden.

```
*** STRING EINLESEN ***
        JSR CHKKOM        ;LOGISCHE FILENUMMER
        JSR GETBYT        ;EINLESEN (X-REGISTER)
        JSR CHKIN         ;EINGABE AUF LOGISCHE DATEI
        LDY #0
INPUT   JSR BASIN
        CMP #13           ;LESEN BIS >>RETURN<<
        BEQ LIESEND       ;UND AB STREND/STREND+1
        STA (STREND),Y    ;IM STRINGSTACK ABLEGEN
        INY
        BNE INPUT         ;UNBEDINGTER SPRUNG (!)
LIESEND STY LENGTH        ;STRINGLAENGE MERKEN
        JSR CLRCH         ;STANDARDGERAETE SETZEN
```

Der Programmteil zum Einlesen des Strings liest zuerst das als Trennzeichen verwendete Komma und die angegebene Filenummer aus dem Basic-Text ein. GETBYT übergibt wie erwähnt die Filenummer im X-Register, in dem sich die Filenummer auch beim Aufruf von CHKIN befinden muß, so daß diese Routine ohne weitere Vorbereitung aufgerufen und die Eingabe auf die geöffnete Datei umgelenkt werden kann.

INPUT verbessert

Das Y-Register wird nun mit Null initialisiert und Zeichen für Zeichen aus der Datei eingelesen und ab dem neuen Anfang des Stringstacks gespeichert, bis das Zeichen RETURN gelesen wird, das das Stringende kennzeichnet. Die in Y enthaltene Stringlänge wird in LENGTH zur späteren Verwendung zwischengespeichert. Durch Aufruf von CLRCH werden wieder die Standardgeräte (Tastatur, Bildschirm) gesetzt.

```
*** POINTER AUF STRINGANFANG ***
     LDA STREND+1     ;POINTR(+1)=
     STA POINTR+1     ;ECHTER STRINGANFANG,
     LDA #255         ;D.H. MOMENTANER
     SEC              ;ANFANG (STREND)
     SBC LENGTH       ;+ DIE DIFFERENZ
     CLC              ;ZWISCHEN DER
     ADC STREND       ;RESERVIERTEN LAENGE
     STA POINTR       ;255 ZEICHEN UND
     BCC NOINC        ;DER TATSAECHLICHEN
     INC POINTR+1     ;LAENGE DES STRINGS
```

Die eingelesenen Zeichen wurden ab der Adresse gespeichert, auf die STREND weist. STREND weist durch die Stringreservierung 255 Zeichen (+2) vor den Beginn des nächsten Strings im Stringstack. Um die unglaubliche Platzverschwendung zu vermeiden, wenn zwar 255 Zeichen reserviert wurden, der eingelesene String tatsächlich jedoch zum Beispiel nur 20 oder 30 Zeichen lang ist, wird nach beendetem Einlesen der String nach oben verschoben, um den Leerraum zwischen dem eingelesenen und dem nächsten String im Stringstack wiederzugewinnen. Dazu wird zuerst anhand der tatsächlichen Stringlänge LENGTH und der daraus resultierenden Differenz zum reservierten Speicherplatz ein Pointer POINTR errechnet, der auf die korrekte Adresse weist, an die der String verschoben werden muß.

```
;*** STRING VERSCHIEBEN ***
NOINC   LDY LENGTH      ;STRING VON:
        DEY             ;STREND BIS STREND+LENGTH
COPY    LDA (STREND),Y  ;NACH:
        STA (POINTR),Y  ;POINTR BIS POINTR+LENGTH
        DEY             ;KOPIEREN
        CPY #255
        BNE COPY
```

Das zeichenweise Kopieren des Strings dürfte problemlos zu verstehen sein. Der String befindet sich an der korrekten Adresse und unsere Routine wäre beendet, doch leider kann das Basic-Programm noch nicht auf den angelegten String zugreifen. Zuvor müssen noch mehrere Pointer korrigiert werden.

```
;*** STRINGDESCRIPTOREN HOLEN ***
        JSR CHKKOM      ;POINTER AUF DIE
        JSR STRPOS      ;DESCRIPTOREN DES
        LDY #2          ;BEIM AUFRUF ANGEGEBENEN
DESCOPY LDA (DESPOI),Y ;STRINGS HOLEN ($47 UND $48)
        STA DESCR,Y     ;UND DIE DESCRIPTOREN
        DEY             ;(LAENGE/ADRESSE(LOW/HIGH))
        BPL DESCOPY     ;NACH DESCR BIS DESCR+2
```

Mit CHKKOM wird das der Filenummer folgende Komma gelesen und STRPOS übergibt in DESPOI ($47 und $48) einen Pointer auf den beim Aufruf angegebenen String. Die Descriptoren dieses Strings werden in einer Schleife nach DESCR (Länge), DESCR+1 (Adresse Low) und DESCR+2 (Adresse high) kopiert.

```
;*** RUECKPOINTER BEHANDELN ***
        LDA DESCR       ;WENN ALTER STRING
                        EXISTIERT
        BEQ ANLEG       ;(DAS HEISST WENN LAENGE
                        <>0),
        TAY             ;WIRD ER UNGUELTIG GEMACHT
        STA (DESCR+1),Y ;(RUECKPOINTER=LAENGE,
        INY             ;RUECKPOINTER+1=$FF)
        LDA #$FF
        STA (DESCR+1),Y
;
ANLEG   LDY LENGTH      ;RUECKPOINTER/RUECKPOINTER
                        +1
        LDA DESPOI      ;WIRD DIE ADRESSE
        STA (POINTR),Y  ;DES LAENGENDESCRIPTORS
        INY             ;ZUGEWIESEN
        LDA DESPOI+1
        STA (POINTR),Y
```

Diesen Programmteil können Sie nicht verstehen, ohne zu wissen, daß der Basic-Interpreter des C 16 am Ende jedes Strings einen Pointer anlegt, der auf den zugehörigen Stringdescriptor, genauer: auf den Längendescriptor des Strings, »zurückweist«. Das heißt, daß nicht nur ein Pointer in der Variablentabelle (die beiden Bytes des Adressendescriptors) auf den String im Stringstack weist, sondern ein weiterer Pointer von diesem String auf die Descriptoren dieses Strings zurückweist. Dieser zusätzliche Pointer hat nur für die Garbage Collection eine Bedeutung, die dadurch erheblich beschleunigt wird.

Entscheidend für uns ist jedoch, daß unsere Routine außer dem eigentlichen String ebenfalls diesen »Rückdescriptor« anlegen muß. Platz dafür ist am Stringende vorhanden, da STRRES den Pointer STREND nicht nur um die angegebene Stringlänge vermindert, sondern – wie bereits von mir angedeutet wurde – um zusätzliche zwei Byte für den ebenfalls anzulegenden Rückdescriptor.

Die Routine »Rückpointer behandeln« prüft zuerst, ob bereits ein String unter dem angegebenen Namen existierte, das heißt ob der Längendescriptor nicht gleich Null ist. Ist dies der Fall, muß der alte String ungültig gemacht werden, da der Basic-Interpreter sonst bei der nächsten Garbage Collection durcheinanderkommt, weil das Low-Byte des Rückpointers die alte Stringlänge und das High-Byte den Wert $FF als Kennzeichen für einen ungültigen String enthält.

Anschließend enthält der neu anzulegende Rückpointer die Adresse der zugehörigen Descriptoren. Er muß auf den Längendescriptor weisen.

```
;*** DESCRIPTOREN/STREND BEHANDELN ***
        LDY #0          ;DESCRIPTOREN AKTUALISIEREN
        LDA LENGTH      ;(LAENGENDESCRIPTOR MIT DER
        STA (DESPOI),Y  ;STRINGLAENGE UND ADRESSEN-
        INY             ;DESCRIPTOREN MIT DER STRING-
        LDA POINTR      ;ADRESSE VERSEHEN
        STA (DESPOI),Y  ;AUSSERDEM STREND/STREND+1
        STA STREND      ;ENTSPRECHEND DER GEAENDERTEN
        INY             ;STRINGADRESSE KORRIGIEREN
        LDA POINTR+1
        STA (DESPOI),Y
        STA STREND+1
        RTS             ;ZURUECK ZUM BASIC
```

Dem Längendescriptor des Strings wird nun die Länge LENGTH und den beiden Adressendescriptoren (Low/High) die geänderte Startadresse POINTR/POINTR+1 zugewiesen. Da der von den reservierten 255 Zeichen unbenutzte Bereich durch das Verschieben des Strings wieder freigegeben wurde, muß natürlich auch STREND/STREND+1 entsprechend der endgültigen Stringadresse korrigiert werden.

Alle Pointer und Descriptoren sind nun völlig korrekt gesetzt und das Basic-Programm kann auf den angelegten String zugreifen, wie das folgende Demoprogramm zeigt:

```
100 REM *** STRING ERZEUGEN ***
110 X$="123456789,"
120 FOR I=1 TO 25:Y$=Y$+X$:NEXT
130 Y$=Y$+"12345"
140 :
150 REM *** STRING SCHREIBEN ***
160 OPEN 1,1,1,"TEST"
170 PRINT#2,Y$
180 CLOSE 2
190 :
200 REM *** STRING LESEN ***
210 OPEN 1,1,0,"TEST"
220 SYS 1630,1,A$
230 CLOSE 2
240 PRINT A$
```

Dieses Demoprogramm schreibt einen String mit einer Maximallänge von 255 Zeichen inklusive dem mit INPUT# keinesfalls einlesbaren Zeichen »,« auf Band und liest ihn mit der vorgestellten Routine in der vollen Länge mit Kommata wieder ein, was für den INPUT#-Befehl völlig unmöglich wäre. Das Basic-Programm kann auf den eingelesenen String wie auf jeden anderen String mit PRINT A$ zugreifen.

Wir sind nun am Ende dieses Artikels angelangt. Nachfolgend finden Sie das Monitorlisting (Listing 1) der Routine, den entsprechenden Datalader (Listing 2) und eine Tabelle, in der alle verwendeten Routinen des Betriebssystems und des Basic-Interpreters aufgeführt sind.

Verzeihen Sie mir bitte, daß die Erläuterungen der Routinen zum Anlegen des Strings, zur Behandlung der Descriptoren und des Rückdescriptors ziemlich knapp ausfielen, aber der Schwerpunkt dieses Artikels war nun einmal nicht der Basic-Interpreter des C 16 und dessen Variablenbehandlung, sondern es sollte gezeigt werden, wie Sie auch in Maschinensprache Daten an beliebige Peripheriegeräten senden oder von diesen empfangen können. (S. Baloui/ah)

Wühlereien im Betriebssystem

Für alle Assembler-Programmierer zeigt der folgende Artikel die Verwendung der wichtigsten ROM-Routinen im C16.

Bevor wir anfangen können, müssen wir noch ein paar Dinge klären.

1) Dieser Artikel geht davon aus, daß Sie Maschinensprachekenntnisse haben.

2) Die vorgestellten Listings werden mit dem integrierten Monitor TEDMON eingegeben. Sie beginnen im Speicher bei $1001, dem Anfang des Basic-Speichers. Damit unsere Programme dort einigermaßen sicher sind, geben Sie bitte im Monitor folgende Befehle ein:

```
>002B 01 11 (setzt Basic-Start herauf)
>1100 00
```

3) Die Programme werden von Basic aus gestartet: `SYS DEC("1001")`

Kernel-Einsprünge

Ab $FF81 stehen im Speicher die sogenannten Kernel-Einsprünge. Diese sind in der Wirkung bei allen Commodore-Computern gleich, was ein eventuell anfallendes Umschreiben (zum Beispiel auf den C 64 oder den Plus/4) erleichtert. Wir besprechen hier in Kurzform die wichtigsten; einige davon kennen Sie schon aus besagtem Artikel im C16-Sonderheft 3, 1986.

VIRES ($FF81)

»VIRES« initialisiert den Video-Chip und den Bildschirm-Editor. Diese Routine wird einfach über »JSR $FF81« aufgerufen und benötigt keinerlei Parameter.

IORES ($FF84)

»IORES« initialisiert die Ein-/Ausgabe-Bausteine des C16. Auch hier sind keine Parameter notwendig.

STMSG ($FF90)

Mit dieser Routine können Sie die Ausgabe von Systemmeldungen (Fehlermeldungen etc.) teilweise oder ganz unterdrücken.

Fehlermeldungen sind Meldungen wie »I/O ERROR #«, Systemmeldungen sind »LOADING« und so weiter. Entscheidend ist der Inhalt des Akkus vor dem Aufruf von »STMSG«:

Akku	Fehlermeldungen?	Systemmeldungen?
$00	ja	ja
$40	nein	ja
$80	ja	nein
$C0	nein	nein

So kann man also alle Meldungen ausschalten:

```
LDA #$C0
JSR $FF90
```

SETLFS ($FFBA)

Diese Routine muß man vor »OPEN«, »LOAD« oder »SAVE« zur Angabe der Fileparameter aufrufen. Sie wurde bereits im Sonderheft 3, 1986 vorgestellt.

SETNAM ($FFBD)

Es gilt das gleiche wie für »SETLFS«.

OPEN ($FFC0)

Diese Routine dient dem Öffnen eines Files, das vorher durch »SETLFS« und »SETNAM« (wie vor »LOAD«) bestimmt wurde. In den Registern übergibt man keine Parameter.

CLOSE ($FFC3)

Gegenstück zu »OPEN«. Damit schließt man ein File wieder. Im Akku muß beim Aufruf die logische Filenummer stehen.

CHKIN ($FFC6)

Mit dieser Routine kann man alle Eingaben von »BASIN« oder »GETIN« statt von der Tastatur von einem vorher geöffneten File erfolgen lassen. Im X-Register übergibt man der »CHKIN«-Routine die logische Filenummer dieses neuen Eingabe-Files.

CKOUT ($FFC9)

Wie »CHKIN«, nur daß das Ausgabegerät für »BSOUT« gesetzt wird. Im X-Register übergibt man die Filenummer der Ausgabedatei.

CLRCH ($FFCC)

Wenn »CHKIN« oder »CKOUT« verwendet wurde, kann man über »CLRCH« wieder die normalen Ein-/Ausgabegeräte (Eingabe: Tastatur; Ausgabe: Bildschirm) einstellen lassen.

BASIN ($FFCF)

Wurde schon im Sonderheft 3, 1986 vorgestellt. Es erfolgt die Eingabe über Tastatur oder dem über »CHKIN« angewählten Gerät.

BSOUT ($FFD2)

Wurde auch schon vorgestellt. Ein Zeichen, das im Akku übergeben wird, wird auf dem Bildschirm oder dem über »CKOUT« eingestellten Gerät ausgegeben. Listing 1 und 2 zeigen Beispiele mit den Ausgabe-Routinen.

LOAD ($FFD5)

Auch nichts Neues. Zu dieser Load-Routine soll an dieser Stelle aber noch ein Hinweis gegeben werden.

Wenn man ein Basic-Programm laden will, muß man:
a) dessen Endadresse dem Computer mitteilen und
b) die Linkpointer (siehe Artikel »Durchblick mit dem Monitor« in diesem Heft) neu berechnen lassen.

Um beides zu bewerkstelligen, läßt man dem Aufruf der

```
. 1001  20 E7 FF JSR $FFE7
. 1004  A9 01    LDA #$01
. 1006  A2 04    LDX #$04
. 1008  A0 00    LDY #$00
. 100A  20 BA FF JSR $FFBA
. 100D  A9 00    LDA #$00
. 100F  20 BD FF JSR $FFBD
. 1012  20 C0 FF JSR $FFC0
. 1015  A2 01    LDX #$01
. 1017  20 C9 FF JSR $FFC9
. 101A  A2 00    LDX #$00
. 101C  BD 2F 10 LDA $102F,X
. 101F  20 D2 FF JSR $FFD2
. 1022  E8       INX
. 1023  E0 08    CPX #$08
. 1025  D0 F5    BNE $101C
. 1027  20 CC FF JSR $FFCC
. 102A  A9 01    LDA #$01
. 102C  4C C3 FF JMP $FFC3

>102F 44 52 55 43 4B 45 52 0D :
```

Listing 1. Beispieltext auf Drucker ausgeben

Load-Routine unmittelbar folgende Befehle folgen:

```
STX $2D    ;Lo-Byte der Endadresse setzen
STY $2E    ;Hi-Byte der Endadresse setzen
JSR $8818  ;Linkpointer berechnen lassen
```

Beim Laden von Maschinenprogrammen ist dies natürlich nicht erforderlich.

SAVE ($FFD8)

Siehe Sonderheft 3, 1986.

STOP ($FFE1)

Mit »JSR $FFE1« fragt man die Stop-Taste ab. Wurde sie gedrückt, ist das Zero-Flag im Prozessor-Statusregister gesetzt, das heißt ein BEQ-Befehl würde ausgeführt werden. Das Programm in Listing 3 wartet, bis die Stop-Taste gedrückt wird.

GETIN ($FFE4)

Siehe Sonderheft 3, 1986.

CLALL ($FFE7)

Diese Routine schließt alle offenen Kanäle. Dabei ist jedoch zu beachten, daß dies nicht wie bei »CLOSE« geschieht. Die entsprechenden Geräte werden von »CLALL« nicht angesprochen, das heißt die Kanäle werden nur innerhalb des Computers geschlossen, während ein File auf Diskette/Kassette für das entsprechende Gerät nach wie vor offenbleibt.

Diese Routine kann man am Anfang eines Programms verwenden, nicht aber als Ersatz für »CLOSE« ($FFC3) betrachten.

PLOT ($FFF0)

Siehe Sonderheft 3, 1986.

Die bisherige Aufzählung sollte nur eine grobe Zusammenfassung sein. Auf einige Routinen gehen wir nun näher ein.

Eine Anwendung der Routinen »CLALL«, »SETPAR«, »SETNAM«, »OPEN«, »CKOUT«, »BSOUT«, »CLRCH« und »CLOSE« ist Listing 1. Zunächst werden alle noch offenen Kanäle geschlossen ($1001). Dann werden 1 als Filenummer, 4 (Drucker) als Gerätenummer und 0 als Sekundäradresse gesetzt ($1004-$100A). Bei $100D/$100F wird dem Computer mitgeteilt, daß wir keinen Filenamen wollen (Länge des Filenamens = 0).

Das File kann geöffnet ($1012) und die Ausgabe des Computers darauf umgeleitet werden ($1015/$1017). Eine kleine Schleife gibt den Text ab $102F über »BSOUT« aus ($101A-$1025). »CLRCH« ($1027) setzt wieder den Bildschirm als Ausgabegerät. Schließlich wird das File geschlossen ($102A/$102C) und das Programm beendet (»JMP $FFC3« entspricht »JSR $FFC3« und »RTS«).

Wie die anderen Beispielprogramme wird Listing 1 mit SYSDEC("1001") gestartet.

Erleichterte Bildschirmausgabe

Wie wir wissen, gibt »BSOUT« das im Akku übermittelte Zeichen auf das aktuelle Ausgabegerät aus; dieses ist der Bildschirm, solange nicht über »CKOUT« ein anderes Ausgabegerät gesetzt ist. Will man gleichzeitig Texte aus dem Bildschirm und das über »CKOUT« eingesetzte Gerät ausgeben, hilft die Routine

BSOUTSCREEN ($DC49)

Diese Routine arbeitet wie »BSOUT«, aber die Ausgabe erfolgt hier in jedem Fall auf dem Bildschirm.

Die Bildschirmausgabe kann auch noch durch weitere Routinen erleichtert werden:

CLEAR ($D88B)

ist eine Abkürzung für

```
LDA #$93    ;$93 = 147 = Code für »Clear Home«)
JSR $FFD2
```

und ist – wie »BSOUTSCREEN« – von »CKOUT« unabhängig.

HOME ($D89A)

ist eine Abkürzung für

```
LDA #$13     ;$13 = 19 = Code für »Home«)
JSR $FFD2
```

und ist wie »CLEAR« von »CKOUT« unabhängig.

SETCR ($D83B)

ist eine Abkürzung für

```
CLC
JSR PLOT
```

und setzt somit den Cursor an die in X und Y angegebene Position.

GETCR ($D849)

ist eine Abkürzung für

```
SEC
JSR PLOT
```

und holt die Cursorposition nach X und Y.

Diese neuen Routinen zur Bildschirmausgabe werden in Listing 2 angewendet, um den Text »(C)64'ER« in die Mitte des Bildschirms zu schreiben.

Ich möchte ausdrücklich darauf hinweisen, daß die Routinen »CLEAR«, »HOME«, »SETCR« und »GETCR« keine Standard-Vektoren sind und somit nur für den C16 gelten.

Jetzt haben Sie die wichtigsten Betriebssystemroutinen kennengelernt. Nicht so bekannt sind vier recht interessante Routinen des TEDMON. Diese Routinen kann man zur hexadezimalen Ausgabe von Zahlen verwenden.

PUTHEX ($FB10)

Im Akku übergeben wir einen Byte-Wert, der dann über JSR $FB10 zweistellig hexadezimal ausgegeben wird.

```
. 1001   20 8B D8 JSR $D88B
. 1004   A2 0A    LDX #$0A
. 1006   A0 0F    LDY #$0F
. 1008   20 3B D8 JSR $D83B
. 100B   A2 00    LDX #$00
. 100D   BD 1C 10 LDA $101C,X
. 1010   20 49 DC JSR $DC49
. 1013   E8       INX
. 1014   E0 08    CPX #$08
. 1016   D0 F5    BNE $100D
. 1018   20 9A D8 JSR $D89A
. 101B   60       RTS

>101C 28 43 29 36 34 27 45 52 :
```

Listing 2. Bildschirmausgabe mit den neuen ROM-Routinen

```
. 1001   20 E1 FF JSR $FFE1
. 1004   D0 FB    BNE $1001
. 1006   60       RTS
```

Listing 3. Abfragen der Run/Stop-Taste

```
. 1001   A9 34    LDA #$34
. 1003   A2 12    LDX #$12
. 1005   20 FF FA JSR $FAFF
. 1008   A9 A0    LDA #$A0
. 100A   20 05 FB JSR $FB05
. 100D   A9 A1    LDA #$A1
. 100F   20 10 FB JSR $FB10
. 1012   A9 2E    LDA #$2E
. 1014   4C D2 FF JMP $FFD2
```

Listing 4. Beispiel zur Anwendung der Routinen des TEDMON

Beispiel:

```
LDA #$D5
JSR $FB10
```

Auf dem Bildschirm erscheint »D5«.

PUTHXS ($FB05)

Diese Routine entspricht »PUTHEX«, allerdings wird nach der Hex-Darstellung des Bytes noch ein Leerzeichen ausgegeben.

MAKHEX ($FB20)

Die Routinen »PUTHEX« und »PUTHXS« bedienen sich dieser Routine.

Im Akku übergibt man »MAKHEX« den umzurechnenden Wert. Man erhält dann im Akku den ASCII-Code der ersten Stelle der zweistelligen Hex-Zahl, im X-Register die zweite Stelle als ASCII-Code zurück.

Beispiel:

```
LDA #$A5
JSR $FB20
```

Im Akku steht dann $41 (ASCII-Code von A), im X-Register $35 (ASCII-Code von 5).

PUTWRD ($FAFF)

Mit dieser Routine ist auch die Ausgabe von 2-Byte-Werten kein Problem. Im Akku wird das Low-Byte, im X-Register das High-Byte der Zahl, die vierstellig hexadezimal ausgegeben werden soll, übergeben.

Listing 4 wendet die aufgezeigten TEDMON-Routinen an.

Der letzte ROM-Einsprung, mit dem wir uns genauer beschäftigen wollen, ist

RESET ($FFF9)

»JMP $FFF9« hat die gleiche Wirkung wie das Drücken der Reset-Taste. Es erfolgt ein Sprung zum Reset-Vektor, wodurch der Einschaltzustand hergestellt wird.

Die Tabelle in Bild 1 zeigt Ihnen noch einmal alle ROM-Routinen zusammengefaßt. Viel Spaß und viel Erfolg bei deren Anwendung. (Florian Müller/ks)

Adresse	Label	Funktion
$8818	LPUPDT	LinkPointer aktualisieren
$D83B	SETCR	Cursor setzen
$D849	GETCR	Cursor holen
$D88B	CLEAR	Bildschirm löschen
$D89A	HOME	Cursor in HOME-Position
$DC49	BSOUTSCREEN	Zeichen auf Bildschirm ausgeben
$FAFF	PUTWRD	2-Byte-Zahl hexadezimal ausgeben
$FB05	PUTHXS	Bytewert hexadezimal mit Space ausgeben
$FB10	PUTHEX	Bytewert hexadezimal zweistellig ausgeben
$FB20	MAKHEX	ASCII-Code der Hex-Darstellung berechnen
$FF81	VIRES	Video-Chip und Editor initialisieren
$FF84	IORES	I/O-Bausteine initialisieren
$FF90	STMSG	Flag für System-/Fehlermeldungen setzen
$FFBA	SETLFS	File-Parameter setzen
$FFBD	SETNAM	File-Namen setzen
$FFC0	OPEN	File öffnen
$FFC3	CLOSE	File schließen
$FFC6	CHKIN	Eingabegerät setzen
$FFC9	CKOUT	Ausgabegerät setzen
$FFCC	CLRCH	Tastatureingabe und Bildschirmausgabe setzen
$FFCF	BASIN	Zeichen von Eingabegerät holen
$FFD2	BSOUT	Zeichen auf Ausgabegerät ausgeben
$FFD5	LOAD	Programm laden
$FFD8	SAVE	Programm speichern
$FFE1	STOP	Stop-Taste abfragen
$FFE4	GETIN	Zeichen eingeben
$FFE7	CLALL	alle offenen Kanäle schließen
$FFF0	PLOT	Cursor setzen (C=0) oder holen (C=1)
$FFF9	RESET	Software-Reset auslösen

Bild 1. Die Routinen und ihre Einsprünge. Die genaue Anwendung der Routinen entnehmen Sie bitte dem Artikel.

Aus klein mach groß

Für Hardware-Freaks ist es relativ einfach, den Speicher des C16/116 auf 64 KByte aufzurüsten. Hier die Bauanleitung dafür.

Es gibt drei Methoden, aus dem kleinen C 16/116 einen großen zu machen:

1. Sie kaufen sich eine Speichererweiterung. Für zirka 130 bis 200 Mark ist sie im Handel erhältlich. Sie wird entweder in den Expansion-Port oder auf die Platine im Computer gesteckt.

2. Die vorhandenen zwei Speicher-ICs (je 8 KByte) bleiben an ihren Plätzen. Auf einer zusätzlichen Platine erweitern Sie den Speicher mit sechs weiteren Speicher-ICs, die ebenfalls je 8 KByte Speicherplatz haben.

Beim C 116 wird dies kritisch, da das Gehäuse voll ausgenutzt ist. Der Verdrahtungsaufwand ist auch erheblich, denn alle Adreß- und Datenleitungen müssen an die Platine gelegt werden. Außerdem wird das ohnehin schwache Netzteil stärker belastet.

3. Die beiden dynamischen RAMs 4416 werden durch dynamische RAMs des Typs 41464 ersetzt. Schaltungsmäßig ist der Aufwand gering.

Diese dritte Möglichkeit wollen wir hier beschreiben. Zunächst müssen wir Sie aber darauf hinweisen, daß beim Öffnen des Gehäuses die Garantie verlorengeht.

Der Umbau beim C 116 und beim C 16 ist im Prinzip gleich, denn die beiden Schaltungen entsprechen einander. Nur die Platinen und die Bezeichnung der Bauteile sind unterschiedlich, wie Sie in Bild 1 (C 116) und Bild 2 (C 16) sehen können. Anhand des C 116 wollen wir den Umbau erklären. Die Änderung läßt sich beim C 116 auch etwas leichter ausführen. Bei Unterschieden beziehen wir uns auf den C 16 in der Klammer. Aber kommen wir nun zur Sache.

Schrauben Sie den Computer auf und ziehen Sie die Verbindungsstecker zur Tastatur und zur LED ab. Nehmen Sie die Platine heraus, nachdem Sie auch diese losgeschraubt haben. Nun werden die beiden ICs U5 und U6 ausgelötet. Da die Leiterbahnen sehr dünn und die Lötpunkte durchkontaktiert sind, können sich dabei Schwierigkeiten ergeben. Wir empfehlen, die IC-Beinchen mit einer Zinnpumpe freizulegen und das IC vorsichtig herauszuheben. Dies gelingt aber nicht immer. Bei manchen Computern sind die Beinchen unten umgebogen. Hier empfiehlt es sich, mit einem spitzen, scharfen Seitenschneider die IC-Beinchen abzuschneiden und dann die noch einzeln stehenden Beinchen mit Pinzette und Lötkolben zu entfernen. Die Platinen-Löcher können Sie anschließend mit Lötkolben und Saugpumpe (notfalls auch Zahnstocher) freimachen. Als nächstes werden hier IC-Fassungen eingelötet. Sie sollten dabei nicht auf den Pfennig achten und gedrehte Fassungen verwenden. Achten Sie darauf, daß die Kerbe der IC-Fassungen am gleichen Ende wie Pin 1 liegt.

Jetzt müssen die Adreßleitungen A14 und A15 an die RAMs gelegt werden. Da die entsprechenden Anschluß-Pins derzeit auf Plus liegen, müssen diese Verbindungen getrennt werden. Die IC-Anschlüsse werden später mit dem Adreßbus verbunden. Im einzelnen ist dazu folgendes zu beachten.

Zwei Unterbrechungen

C 116: Am IC U8 (Bild 1) muß auf der Bauteileseite die Verbindung zwischen Pin 14 und Pin 16 mit einer scharfen Klinge (Skalpell, Teppichmesser) unterbrochen werden.

C 16: Am IC U8 (Bild 2) muß auf der Lötseite die Verbindung von Pin 14 zur dicken Plusleitung unterbrochen werden.

C 116: Am IC U7 (Bild 1) muß auf der Lötseite die Verbindung zwischen Pin 2 und Pin 16 unterbrochen werden.

C 16: Am IC U7 (Bild 2) muß auf der Bauteileseite unter dem IC die Verbindung zwischen Pin 2 und Pin 16 unterbrochen werden. Ein spitzes Skalpell eignet sich gut dafür. Eine weitere Möglichkeit ist, den Pin 2 des ICs mit einem Seitenschneider so weit unten an der Platine wie möglich abzuschneiden und hochzubiegen.

Auf alle Fälle sollten Sie sicherheitshalber mit einem Durchgangsprüfer oder einem Ohmmeter überprüfen, ob die Leitungen wirklich unterbrochen sind. Dabei ist jedoch folgendes zu beachten. Wenn der Pluspol des Meßgerätes am Pin 16 liegt, wird ein Widerstand unter 1000 Ohm gemessen. Also, den Minuspol des Meßgerätes an Pin 16, dann muß eine

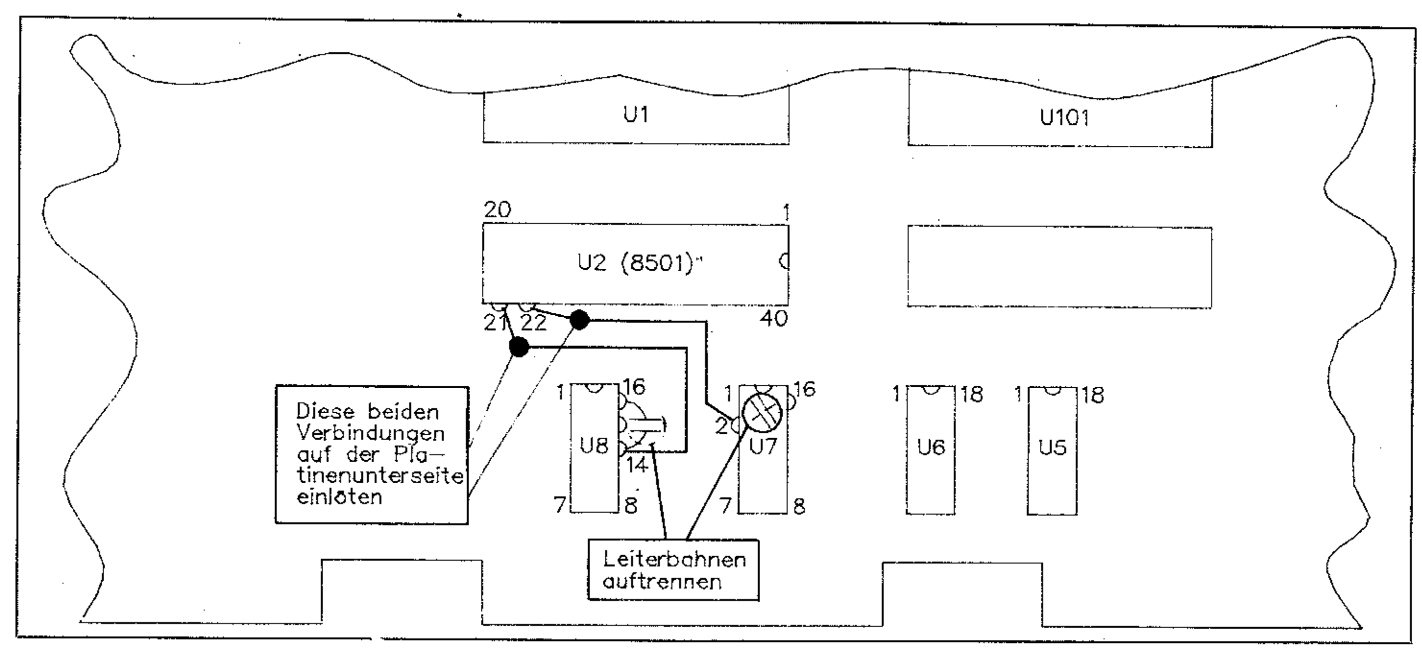


Bild 1. Die Bestückungsseite der C116-Platine. Die notwendigen Unterbrechungen und Verbindungen sind eingetragen.

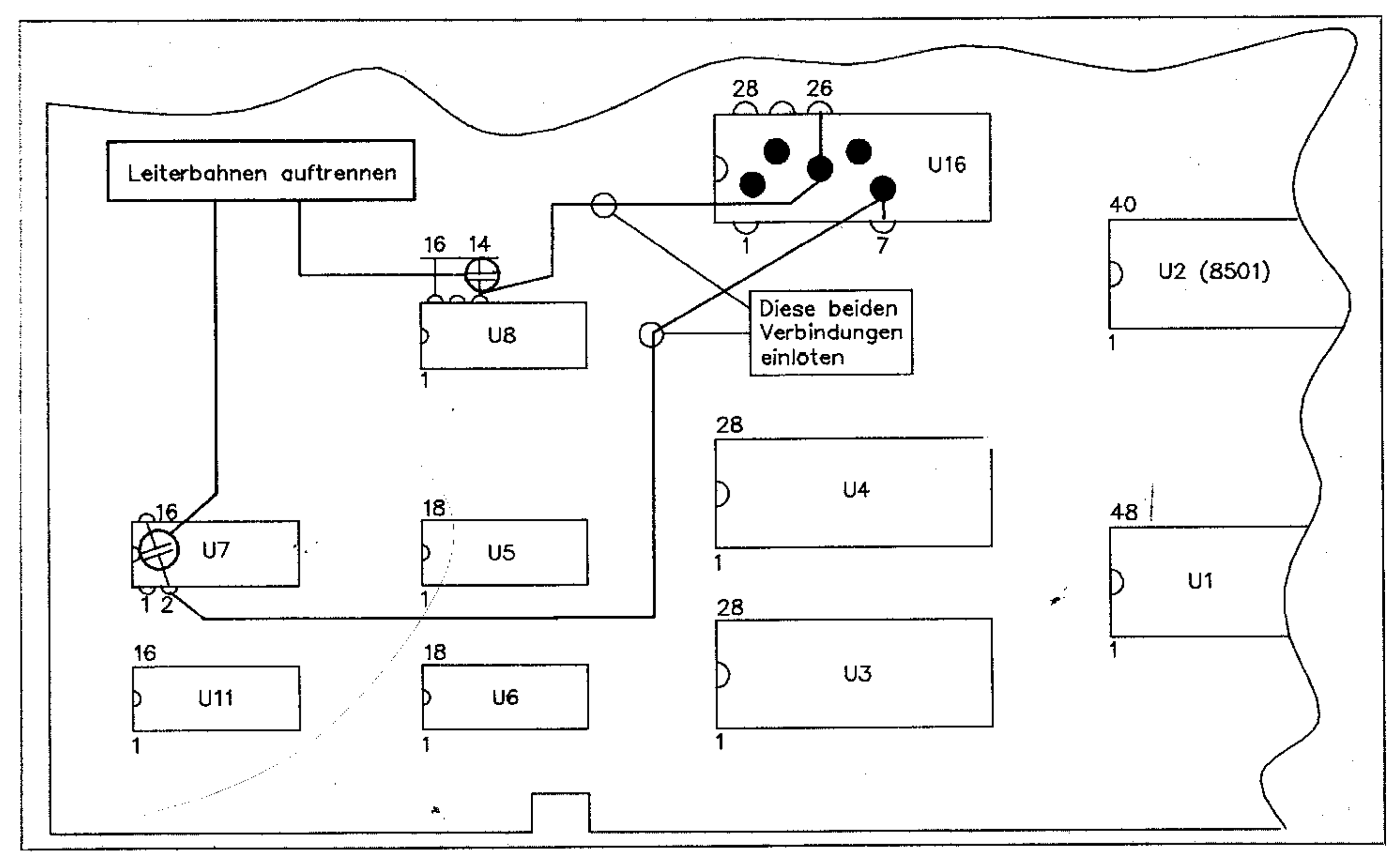


Bild 2. Die Bestückungsseite der C16-Platine. Auch hier sind die beiden Unterbrechungen und Verbindungen eingetragen.

eindeutige Unterbrechung angezeigt werden (Hinweis: manche Meßgeräte haben bei der Widerstandsmessung Plus und Minus vertauscht).

Messung: Am IC U7 zwischen Pin 14 und 16, am IC U8 zwischen Pin 2 und 16.

C 116: Suchen Sie jetzt das IC U2 (Prozessor 8501) und dort die Pins 21 und 22. Sie müssen ohne jede Verbindung sein.

C 16: Suchen Sie sich das IC U16 und dort die Pins 7 und 26. Jeweils parallel dazu sind Durchkontaktierungen als Lötpunkte (Bild 2).

Kennzeichnen Sie alle 4 Anschlüsse mit Filzschreiber, je einen am U7 und U8, zwei am U2 (U16 beim C 16). Jetzt müssen die Anschlüsse der RAMs mit kurzen Drähten an die Adreßleitungen A14 und A15 gelegt werden. Beim C 116 sind die kürzesten Verbindungen auf der Platinenunterseite von U7 Pin 2 nach U2 Pin 22 und von U8 Pin 14 nach U2 Pin 21. Beim C 16 ist die kürzeste Verbindung, wenn Pin 2 von U7 abgeschnitten wurde, vom Pin 2 seitlich unter dem Sockel des IC U16 hindurch bis an die Durchkontaktierung (eventuell das IC dazu vorsichtig heraushebeln).

Zwei neue Verbindungen

Wenn Sie die Leiterbahn unterbrochen haben, können Sie beide Verbindungen auf der Lötseite machen: von U7 Pin 2 nach U16 Pin 7 und von U8 Pin 14 nach U16 Pin 26, jeweils an der Durchkontaktierung anlöten.

Im C 16 entspricht das IC U16 dem U101 des C 116. Da U2 beim C 16 weiter entfernt liegen als die Durchkontaktierungen, sind die vorgeschlagenen Punkte die günstigsten.

Die neuen RAM-ICs (siehe Stückliste) brauchen Sie nur noch in die IC-Sockel stecken und eine Computerplatine mit 64 KByte RAM liegt vor Ihnen.

Jetzt bleibt lediglich übrig, alles nochmals genau zu überprüfen.

Hier noch einige Tips für den Zusammenbau. Der Tastaturstecker kann nicht falsch aufgesteckt werden, weil der 2. Pin von links als Verriegelung dient.

Beim C 116 muß aus der Metallabschirmung für die ICs U5 und U6 ein Stück ausgeschnitten werden. Die Funktion wird aber dadurch nicht beeinflußt. Beim C 16 gibt es dank der Abschirmfolie keine Probleme dieser Art.

Neue RAMs rein

Dem Leuchtdiodenstecker macht ein Verdrehen nichts aus, weil beide äußeren Pins miteinander verbunden sind.

Wenn Sie den Computer wieder zusammengeschraubt haben, brauchen Sie ihn nur noch anzuschließen. Und jetzt kommt der spannende Moment: einschalten und – 60671 Byte free.

Die Kosten für den Umbau betragen je nach IC-Typ und Händler zwischen 35 und 60 Mark.

Für alle, die sich den Umbau nicht selbst zutrauen, haben wir im Aktuell-Teil unter »Speichererweiterung einbauen lassen« zwei Adressen angegeben, wo Sie den Umbau für zirka 100 Mark in Auftrag geben können. (Manfred Velt/kn)

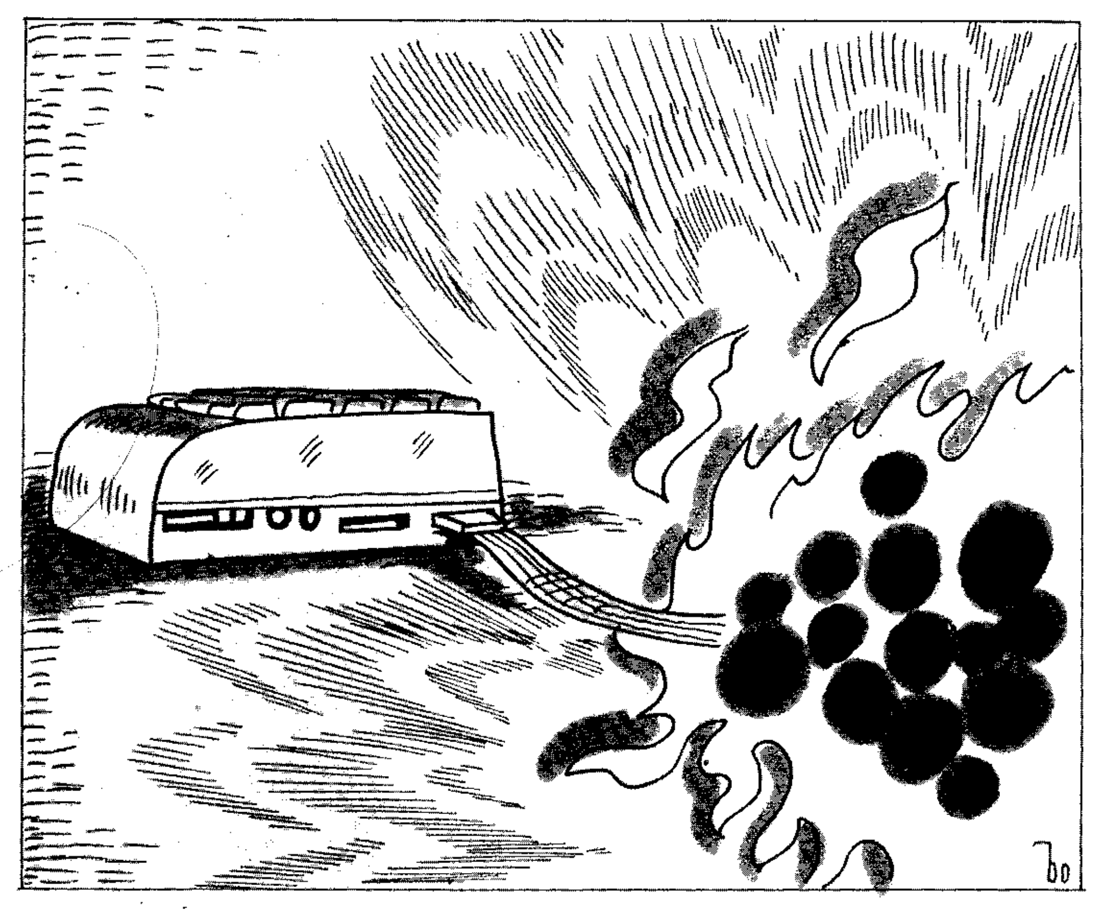
Stückliste

2 isolierte Drähte je 5 cm
2 IC-Sockel 18polig, gedreht
2 IC Nec µpd 41464 C15
(Erhältlich bei: Frank Elektronik GmbH, Postfach 840073, Mathiasstraße 3, 8500 Nürnberg 84, Tel. 0911/327732

Ersatztypen: Fujitsu MB 814 64-15
TMS 4464 – 15 N 2
Beide Ersatztypen haben wir jedoch nicht testen können.

Tabelle. Die benötigten Bauteile

Der heiße Draht nach draußen



Da der C16/116 keinen User-Port besitzt, kommt dem Expansion-Port eine besondere Bedeutung zu. Hier finden Sie eine Übersicht zur Kontaktbelegung.

Natürlich ist der Expansion-Port beim C16/116 und Plus/4 einmal wieder völlig anders als beim C64. Nicht nur, daß die Anschlußbuchse eine ganz andere Bauform und damit auch einen anderen Abstand zwischen den Kontakten hat, nein, auch die Belegung der einzelnen Anschluß-Pins ist teilweise vollkommen anders. Außerdem stehen Ihnen beim C64 sechs Anschlüsse weniger zur Verfügung. Einen direkten Vergleich erhalten Sie, wenn Sie die nachstehende Tabelle neben das »64'er Extra« aus der Ausgabe 4/86 legen. Dort ist in ähnlicher Weise der Expansion-Port des C64 beschrieben.

Nun aber zu unseren drei Computern. Zum Glück sind die Expansion-Ports beim C16, C116 und Plus/4 identisch. Einige Besonderheiten wollen wir Ihnen auch noch näher erläutern.

Da ist zunächst der Zugriff auf RAM-Bausteine (löschbare Speicher). Er wird durch die Signale $\overline{RAS}$, $\overline{CAS}$ und MUX geregelt. Da die im C16 verwendeten RAMs nur über insgesamt 16 Anschlüsse verfügen, sind sie für die Adressierung in Reihen (row) und Spalten (column) aufgeteilt. In Bild 1 sehen Sie ein Beispiel mit 3 Reihen und 3 Spalten. Aufgrund der wenigen Anschlüsse müssen bei den RAMs Reihen und Spalten nacheinander adressiert werden. Dies zeitlich zu organisieren ist die Aufgabe der Signale $\overline{RAS}$, $\overline{CAS}$ und MUX.

Ferner verfügen der C16/116 und Plus/4 über ein sogenanntes Bank-Switching. Das bedeutet, es können verschie-

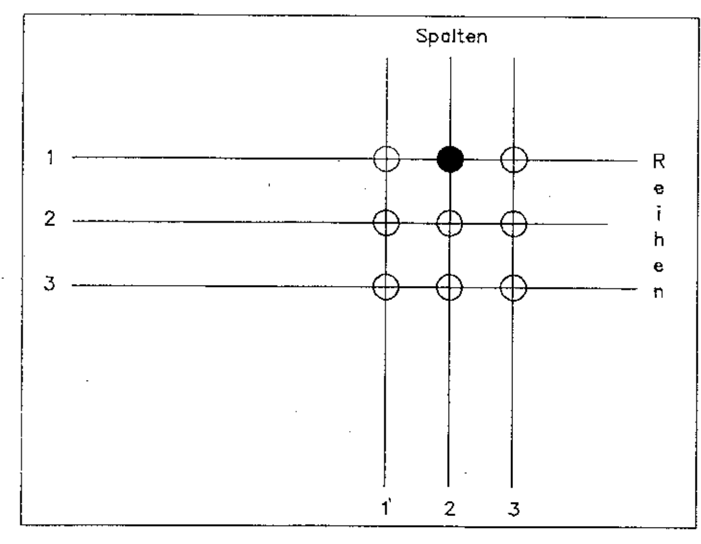


Bild 1. Adressierung von Speicherzellen über Reihen und Spalten. Die schwarz markierte Speicherzelle ist über die Reihe 1 und Spalte 2 angesprochen.

dene Programme über Speicherbänke eingeblendet werden. Über die Signale $\overline{\text{CS0}}$ und $\overline{\text{CS1}}$ können zwei verschiedene Speicherbereiche eingeblendet werden, und zwar die Bereiche $8000 bis $BFFF (dezimal 32768-49151) und $C000 bis $FFFF (dezimal 49152-65535). Welche Speicherbank für die Einblendung angesprochen werden soll, wird durch die Signale C1 LOW, C1 HIGH, C2 LOW und C2 HIGH bestimmt. Über die Adressen $FDD0 bis $FDDF (dezimal 64976-64991) können Sie die Bänke vorgeben.

Abschließend noch zwei Hinweise: Bild 2 zeigt den Expansion-Port, wie er von hinten betrachtet am Computer zu sehen ist. Die Zahlen gelten also für die oberen Kontakte und die Buchstaben für die unteren.

Häufig finden Sie einen Strich über der Signalbezeichnung. Hier handelt es sich um sogenannte »Nicht-Anweisungen«, die auch als Low-aktiv bezeichnet werden. Das bedeutet, wenn diese Leitungen auf Masse (Low) gelegt werden, ist die entsprechende Funktion aktiv. (J. Sahlmann/kn)

25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

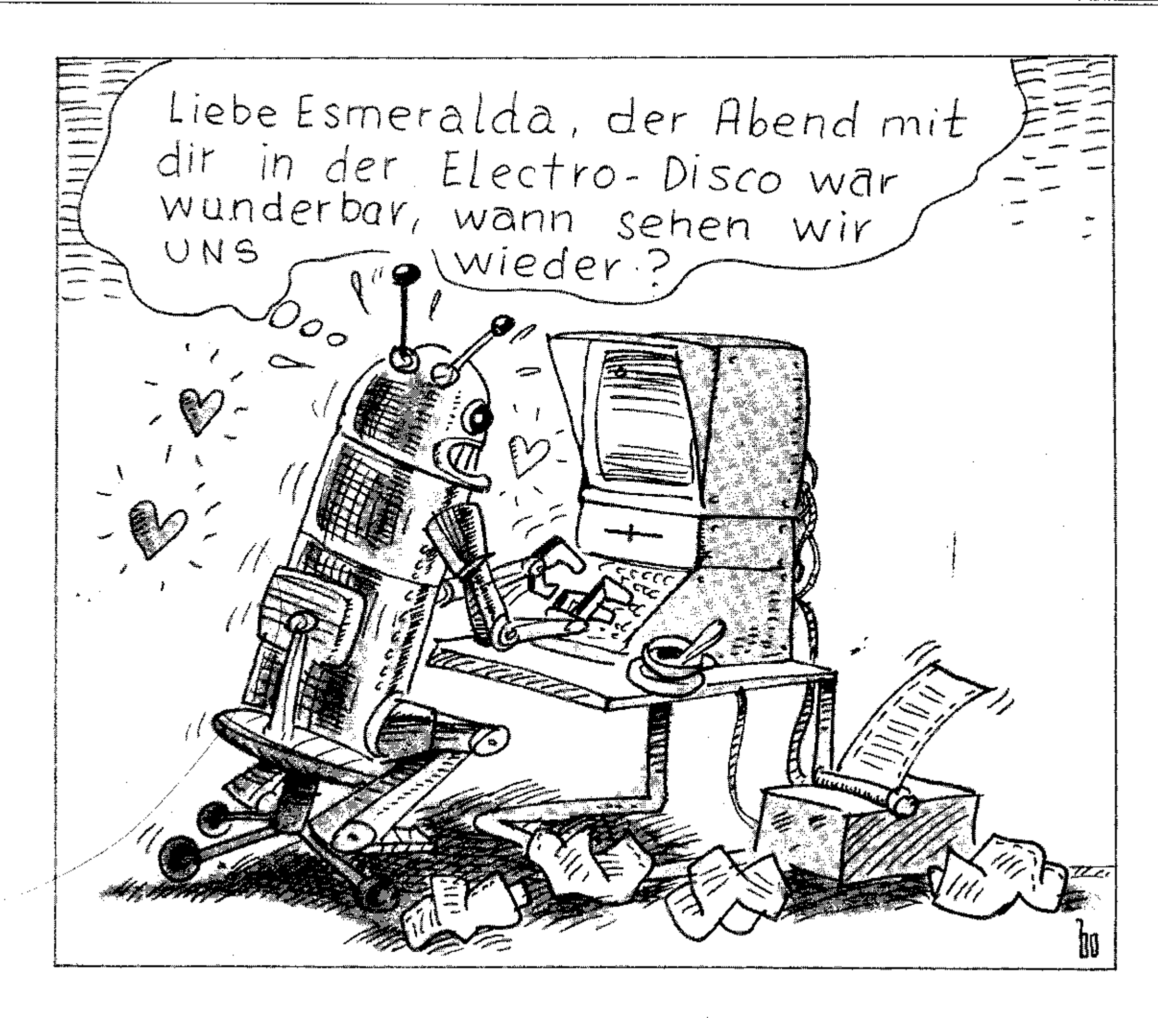
CC BB AA Z Y X W V U T S R P N M L K J H F E D C B A

Bild 2. Der Expansion-Port des C16, C116, Plus/4 – von hinten gesehen

PIN	Name	Beschreibung	Bemerkung
1	GND	Systemmasse	
2	+5 V DC	Betriebsspannung	
3	+5 V DC		
4	$\overline{\text{IRQ}}$	Interrupt-Anforderung	Wenn $\overline{\text{IRQ}}$ auf Masse (GND) gelegt wird, arbeitet der Prozessor den momentanen Befehl ab und verzweigt in die Interrupt-Routine des Betriebssystems
5	R/$\overline{\text{W}}$	Lesen/Schreiben	Bei jedem Lesezyklus (z.B. LDA,LDX) wird diese Leitung auf +5 V gesetzt, bei jedem Schreibzyklus (z.B. STA) auf 0 V (= Masse)
6	C1 HIGH	Signal für Bank-Switching	Steuerleitungen für das Bank-Switching (siehe auch Pin B und Text)
7	C2 LOW	Signal für Bank-Switching	
8	C2 HIGH	Signal für Bank-Switching	
9	$\overline{\text{CS1}}$	Chip-Select-Leitung	Einblendung externer Bausteine (RAM oder ROM) in die Speicheradressen $ 8000 - $ BFFF (= 16 KByte)
10	$\overline{\text{CS0}}$	Chip-Select-Leitung	Einblendung externer Bausteine (RAM oder ROM) in die Speicheradressen $ C000 - $ FFFF (= 16 KByte)
11	$\overline{\text{CAS}}$	Column Address Select	Speicherzugriff auf das RAM
12	MUX	Multiplex-Signal	(siehe auch im Text)
13	BA	Buszugriff vom TED	BA liegt auf +5 V, wenn der TED (Baustein für Ton, Bild und I/O) auf Daten- oder Adreßbus zugreift. Liegt BA auf +5 V liegt , so dürfen keine externen Bausteine auf den Bus zugreifen.
14	D7	Datenbus Bit 7	Datenbus (ungepuffert)
15	D6	Datenbus Bit 6	
16	D5	Datenbus Bit 5	
17	D4	Datenbus Bit 4	
18	D3	Datenbus Bit 3	
19	D2	Datenbus Bit 2	
20	D1	Datenbus Bit 1	
21	D0	Datenbus Bit 0	
22	AEC	Address Enable Control	Wenn AEC auf Masse liegt, ist der Adreßbus am Prozessor hochohmig
23	EXT AUDIO	Externe Audioleitung	An EXT AUDIO liegt dasselbe Tonsignal wie an Pin 5 des Videoausgangs

PIN	Name	Beschreibung	Bemerkung
24	Φ 2	Systemtakt	Takt von Baustein U3 (1.8432 MHz)
25	GND	Systemmasse	
A	GND	Systemmasse	
B	C1 LOW		Steuerleitung für das Bank-Switching (siehe auch Pins 6,7,8)
C	$\overline{\text{RESET}}$	Computer Kaltstart	Liegt RESET auf der Masse, so löst der Computer einen Reset aus
D	$\overline{\text{RAS}}$	Row Address Select	Speicherzugriff auf das RAM (siehe Text)
E	Φ 0	Systemtakt	Bei fallender Flanke sind Daten gültig
F	A15	Adreßbus Bit 15	Adreßbus (ungepuffert)
H	A14	Adreßbus Bit 14	
J	A13	Adreßbus Bit 13	
K	A12	Adreßbus Bit 12	
L	A11	Adreßbus Bit 11	
M	A10	Adreßbus Bit 10	
N	A 9	Adreßbus Bit 9	
P	A 8	Adreßbus Bit 8	
R	A 7	Adreßbus Bit 7	
S	A 6	Adreßbus Bit 6	
T	A 5	Adreßbus Bit 5	
U	A 4	Adreßbus Bit 4	
V	A 3	Adreßbus Bit 3	
W	A 2	Adreßbus Bit 2	
X	A 1	Adreßbus Bit 1	
Y	A 0	Adreßbus Bit 0	
Z	-	nicht angeschlossen	
AA	-	nicht angeschlossen	
BB	-	nicht angeschlossen	
CC	GND	Systemmasse	

Der Expansions-Port der Computer C16, C116 und Plus/4



Text-Manager

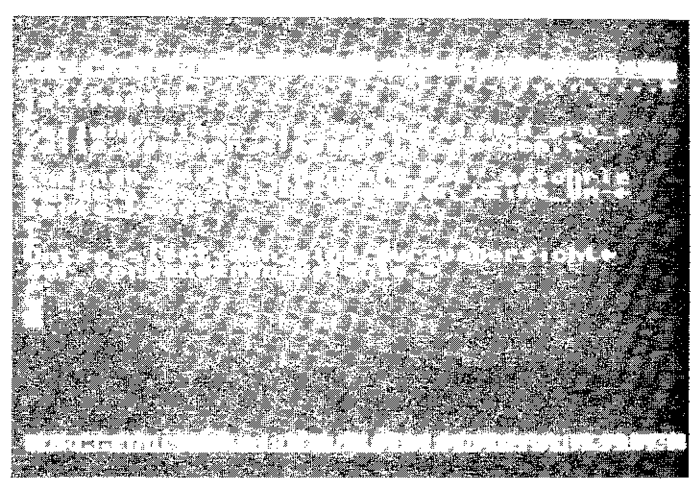
Briefe schreiben leichtgemacht

Der C16/116 gewann in letzter Zeit seines niedrigen Preises wegen zunehmend an Beliebtheit. Die Nachfrage stieg plötzlich enorm an. Warum soll man den C16/116 nicht als intelligente Schreibmaschine verwenden? Eine Lösung bietet das Programm Text-Manager.

Mit einem C16-Boom hatte kein Software-Produzent gerechnet. Nur einige Spiele kamen auf den Markt. Viele C16-Besitzer wollten ihren C16 aber nicht nur als Spielcomputer, sondern ebenso für ernsthafte Anwendungen nutzen. Eines der wichtigsten und begehrtesten Anwendungsprogramme für Heimcomputer ist wohl ein Textverarbeitungssystem. Mit solch einem Programm können Texte beliebiger Art auf einfache Weise erstellt, editiert und in einer schönen Form ausgedruckt werden.

Der C16/116-Text-Manager ist ein gelungenes Textverarbeitungssystem, bei dem eine gute Kombination aus wenig Speicherbedarf und relativ viel Komfort erzielt wurde.

Das Programm ist sehr speicherplatzsparend aufgebaut und nutzt wirklich jedes verfügbar Byte, um möglichst viel Textspeicher zur Verfügung stellen zu können.



Der Text-Manager enthält die wichtigsten Funktionen eines Textprogramms

Nach dem Start des kurzen (zirka 4,25 KByte) Programmes stehen noch zirka 7900 Byte (7900 Zeichen) für den Text bereit.

Obwohl das Programm so kurz ist, bietet es erstaunlich viel Bedienungskomfort. Es wurden viele wichtige Funktionen und Kommandos integriert.

Der Editor ist sehr leicht zu bedienen. Der Text wird einfach endlos eingetippt. Dabei haben die Cursor-Tasten die gleiche Funktion wie im Basic-Editor; ebenso die Tasten <HOME>, <DEL> und <INST>. Die <CLEAR>-Taste setzt den Cursor allerdings in die letzte Textzeile.

Der Editor beherrscht das sogenannte »Wordwrapping«. Darunter ist zu verstehen, daß angefangene Wörter, die nicht mehr in eine Zeile passen, automatisch in die nächste Zeile verschoben werden. Es kann also einfach drauflos geschrieben werden, ohne sich um das Zeilenende zu kümmern. Lediglich wenn ein Absatz erzeugt werden soll, ist die <RETURN>-Taste zu drücken.

Komfortabler Editor

Die Funktionstasten wurden mit nützlichen Editierfunktionen belegt. Insgesamt stehen dem Benutzer 10 wichtige Kommandos zur Verfügung.

Der Befehl »Bytes Free« dient zur Anzeige des für die Texteingabe verfügbaren Speicherplatzes. Der Text-Manager geht sehr sparsam mit dem Speicherplatz um. Wordwrapping benötigt keinen zusätzlichen Speicherplatz. Eine ganze Leerzeile wird ebenso wie ein Textzeichen nur als ein einziges Byte gespeichert.

Mit »Width« läßt sich die Textbreite zwischen 35 und 99 Spalten (Zeichen pro Zeile) variieren. Der Standardwert ist auf 70 Spalten voreingestellt. Dabei wird der Text aber nicht etwa als 40-Spalten-Text dargestellt, sondern der Text-Manager arbeitet mit Scrolling in alle vier Richtungen. Der Bildschirm ist quasi ein Fenster, durch das man immer einen Textausschnitt sieht. Das Textfenster wird dabei je nach Stellung des Cursors über den Text verschoben. Bei einer Textbreite von weniger als 38 Spalten findet kein horizontales Scrollen mehr statt.

Will man im Text eine bestimmte Zeichenkette suchen, so steht dem Benutzer der »Find«-Befehl zur Verfügung. Nach dem Aufruf des »Find«-Befehls kann das gesuchte Wort, beziehungsweise die Zeichenkette eingegeben werden. Nach Betätigung der <RETURN>-Taste sucht der Text-Manager im Text ab der Cursorposition nach dem gewünschten Ausdruck. Wurde eine Textstelle gefunden, kann durch wiederholtes Drücken der <RETURN>-Taste nach einer weiteren Textstelle gesucht werden.

Die »Replace«-Anweisung arbeitet ähnlich wie »Find«. Mit diesem Befehl kann jedoch die gesuchte Textstelle automatisch durch eine zweite Zeichenkette ersetzt werden. Der Benutzer muß allerdings die Veränderung mit der <RETURN>-Taste bestätigen. Im Gegensatz zum »Find«-Befehl verlangt »Replace« bei der Eingabe zwei Zeichenketten, die zu Suchende und die Ersetzende.

Um eine Zeichenkette global im gesamten Text zu ersetzen (ohne Bestätigung durch <RETURN>), gibt es einen speziellen Replace-Befehl. Mit dem Befehl »Disc Command« können alle Diskettenbefehle zur Floppy gesandt werden. Für Datasetten-Benutzer erübrigt sich dieses Kommando. Außer den Befehlen für Formatieren, Initialisieren, Löschen und Validieren kann zusätzlich durch Eingabe des »$«-Zeichens das Directory der Diskette gelesen werden.

Das Kommando »Save« dient zum Speichern des Textes auf Kassette oder Diskette. Zuvor verlangt das Programm noch die Eingabe eines Textnamens. Dieser muß allerdings nur im Falle einer Speicherung auf Diskette angegeben werden.

Um einen Text wieder zu laden, muß die »Load«-Anweisung angewählt werden. Auch hier kann beim Laden von Kassette der Textname weggelassen werden.

Das Kommando »Print« ist für ein Textverarbeitungsprogramm mit Sicherheit der wichtigste Befehl. Damit läßt sich der eben erstellte Text endlich aufs Papier bringen. Leider können im Drucker-Menü nur 3 Parameter verändert werden.

1) Einzelblatt oder Endlospapier
2) Papierlänge
3) Druckzeilen pro Blatt.

Das reicht dem durchschnittlichen Anwender in der Regel. Wünschenswert wäre aber zumindest die Veränderung der Startspalte (der linke Rand) beim Ausdrucken.

Der Text-Manager druckt den Text im Blocksatz aus, das heißt, der durch das Wordwrapping erzeugte Flatterrand wird ausgeglichen, indem eine Zeile durch Einfügen von Leerzeichen zwischen den einzelnen Worten aufgefüllt wird. Der Ausdruck ist sozusagen links- und rechtsbündig, wie der Text in diesem Sonderheft. Es ist noch zu bemerken, daß der Text-Manager ausschließlich im Blocksatz druckt. Ein gewollter Flatterrand ist nicht einstellbar.

Schwierigkeiten treten auf, will man Steuercodes an Centronics-kompatible-Drucker senden. Diese Möglichkeit wurde beim C 16-Text-Manager völlig außer acht gelassen.

Wegen des geringen Speicherplatzes des C 16 ist man eben zu Einschränkungen gezwungen. Mit Textverarbeitungssystemen wie Vizawrite oder Master-Text für den C 64 läßt sich der Text-Manager nur bedingt vergleichen.

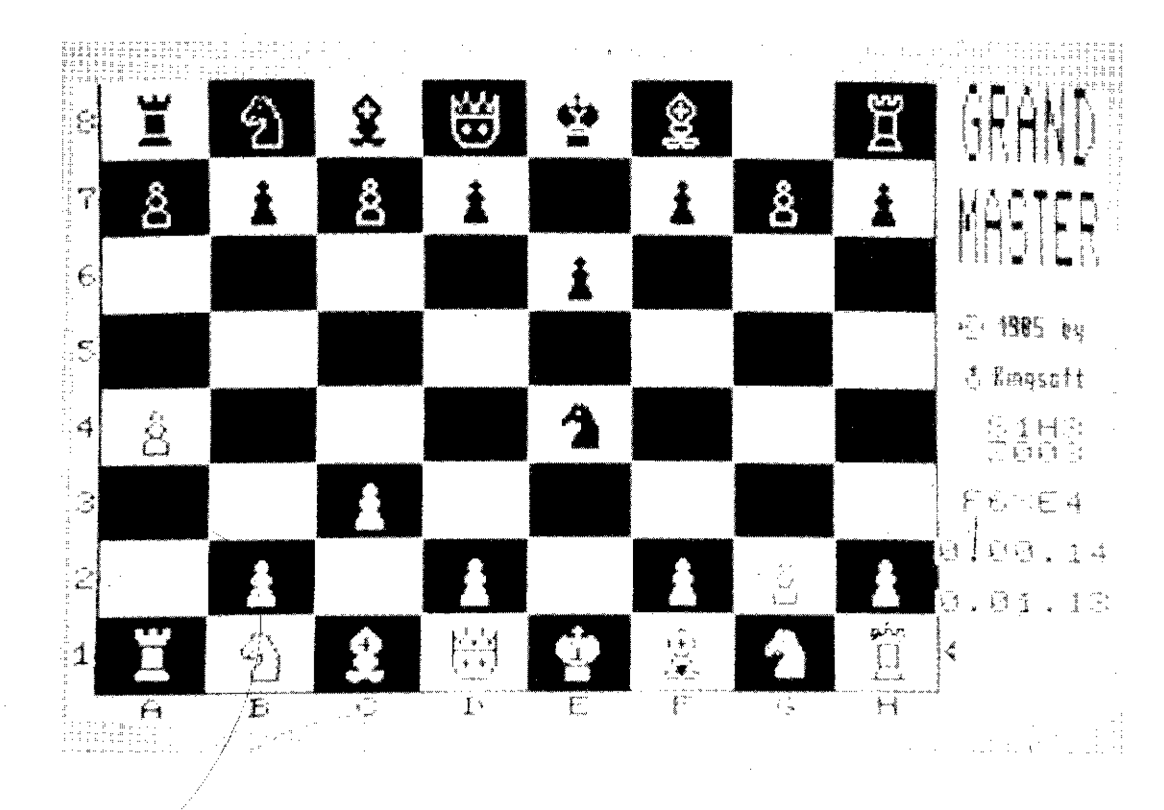
Für Diskette und Datasette

Der letzte Befehl »Quit« dient zum Verlassen des Programms. Aus Sicherheitsgründen muß diese Wahl noch mit <RETURN> bestätigt werden. Alle Kommandos sind bequem über <CTRL> + Anfangsbuchstabe des Befehls zu erreichen.

Für den C 16/116 ist die Anschaffung des Text-Managers durchaus lohnenswert, wenn man sich bewußt ist, daß man vom C 16 nicht die Leistungen erwarten darf als beispielsweise vom C 64.

(Christian Quirin Spitzner/hm)

Info: Markt&Technik-Verlag, Hans-Pinsel-Str. 2, 8013 Haar, 089/4613-0, Preis: 49 Mark (mit Kassette).



Der C 16 als Spielcomputer

Der Erfolg der Heimcomputer in den letzten Jahren wurde nicht zuletzt durch eine Flut von interessanten Spielen ausgelöst. Welche Spiele lohnen sich für den C16?

Viele unserer Leser gehören sicherlich zu der Personengruppe, die, vom preiswerten Angebot einer bekannten Supermarktkette überwältigt, einen Heimcomputer gekauft haben und sich nach der Durcharbeitung des Basic-Kurses fragen, was sie nun eigentlich mit der Kiste anfangen sollen. Ein beliebtes Anwendungsgebiet für alle Heimcomputer sind die Videospiele, die inzwischen Tausenden von jüngeren und älteren Computerbesitzern über verregnete Nachmittage hinweghelfen. Um Ihnen den Einstieg in die Welt der Spiele zu erleichtern, wollen wir Ihnen einige gute Programme vorstellen.

Spiele aus deutschen Landen...

Doch vorher noch einige allgemeine Worte. Alle Spiele, die wir getestet haben, wurden für den C 16 und C 116 programmiert und laufen im allgemeinen auch problemlos auf dem Plus/4. Fast alle Spiele lassen sich über die Tastatur steuern, mit einem Joystick hat man jedoch ein »leichteres Spiel«. Im übrigen darf man vom C 16 keine grafischen Bildschirm-Orgien wie beim C 64 erwarten. Aufgrund des geringen Speicherplatzes und der wenigen Möglichkeiten des Video-Chips (so gibt es beispielsweise keine Sprites) ist der C 16 nicht der ideale Spielecomputer. Ebenso verhält es sich beim Sound. Trotzdem ist es erstaunlich, was manche Programmierer aus dem Gerät herauskitzeln können.

In Deutschland bemüht sich die Firma Kingsoft sehr um die C 16-Besitzer. So gibt es auch einen ganzen Stapel Spielprogramme der unterschiedlichsten Stilrichtungen. Neben Action- und Geschicklichkeitsspielen fällt besonders das Schachprogramm »Grandmaster« (siehe oben) auf. Dieses recht spielstarke Programm kann auch durch seine ansprechende Grafik beeindrucken. Wer sich nicht extra einen Schachcomputer kaufen möchte, kann für wenig Geld seinen C 16 zum Schachpartner machen.

»Grandmaster« wird übrigens in einer Spielesammlung mit drei weiteren Spielen (»Tom«, »Galaxy« und »Ghost Town«) verkauft. Die drei anderen Programme sind dabei mehr dem Action- und Geschicklichkeitsgenre zuzuordnen.

Kingsoft bietet aber auch andere interessante Spiele an. Eines der komplexesten ist das »Bongo Construction Set« (Bild 3). Bongo die Supermaus muß eine Prinzessin befreien. Zu diesem Zweck müssen auf einem Gerüst verschiedene Gegenstände eingesammelt werden. »Bongo« gehört damit zur Gruppe der Jump-And-Run-Spiele (Springen und Laufen). Langweilig wird das Spiel dabei so schnell nicht, da Sie eigene Spielfelder herstellen können. Damit ist »Bongo Construction Set« nicht nur ein Spiel, sondern ein richtiger Baukasten für Spiele, der für monatelange Unterhaltung sorgen kann.

Wer lieber knallharte Action möchte, der sollte sich »Legio-

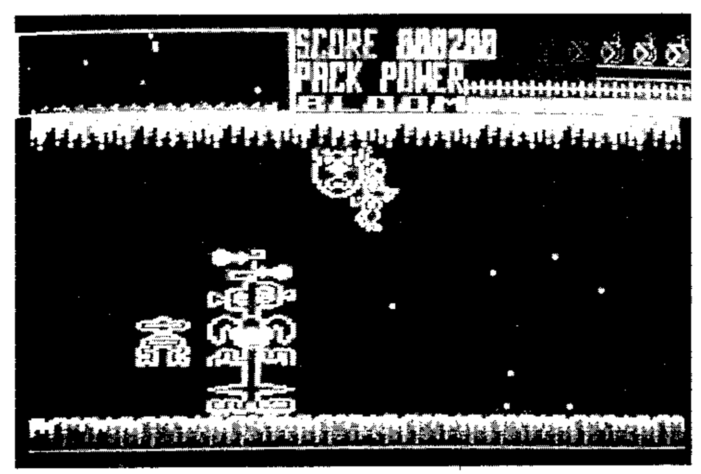
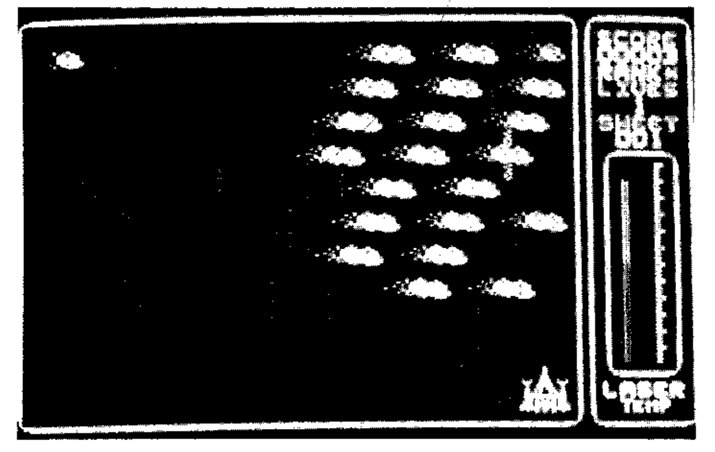


Bild 1. Petals of Doom


Bild 2. Xargon Wars

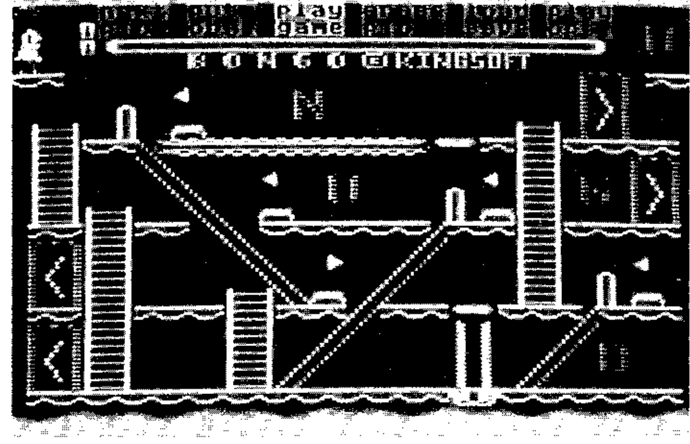


Bild 3. Bongo Construction Set

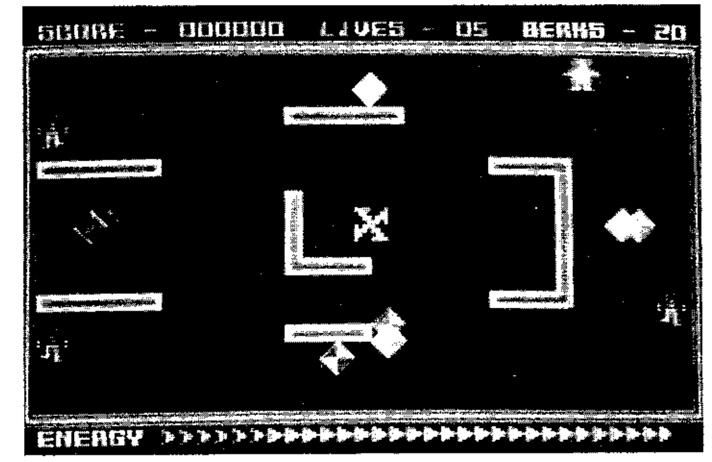


Bild 4. Berks Trilogie

naire« ansehen. Hier kämpft sich ein Bildschirmsöldner durch dichten Dschungel und tritt gegen wahre Hundertschaften von feindlichen Soldaten an. Das Spiel erinnert an den Spielhallen-Knüller »Commando«. Von »Commando« gibt es zwar auch eine C 16-Version, die aber nicht viel mit dem Automatenspiel gemeinsam hat.

Wieder etwas mehr futuristisch geht es bei »Space Pilot« zu, bei dem der Spieler sich wieder gegen fremde Raumschiffe verteidigen muß, ein Thema, das es anscheinend in unendlich vielen Variationen für Computer gibt.

Damit ist das Angebot von Kingsoft noch lange nicht erschöpft, aber leider reicht hier der Platz nicht, um noch mehr Spiele dieser Firma vorzustellen. Schließlich gibt es auch noch einige englische Firmen, die C 16-Software produzieren. Deren Programme werden zum größten Teil von der Firma Rushware vertrieben.

...und aus England

Einer der englischen Software-Produzenten ist Gremlin Graphics, von denen es ebenfalls eine interessante Spielesammlung namens »C 16 Classics« gibt. Auf dieser kann man die Spiele »Tycoon Tex«, »Dorks Dilemma«, »Xargon Wars« und »Petals of Doom« finden. Alle vier Spiele sind sehr actionreich und bieten tolle grafische Effekte, wie Laufschriften, schnelles Scrolling und Animation der Spielfiguren. »Xargon Wars« (Bild 2) ist eine Variante des »Space Invader«-Themas: Der Spieler muß mit seinem Raumschiff die Invasoren aus dem Weltall vernichten, die in verschiedenen Formationen angreifen. »Petals of Doom« (Bild 1) wiederum erinnert an den Spielhallenklassiker »Defender«: Ein recht schlagkräftiger Gärtner muß die Pflanzen eines futuristischen Gartens bewachen. Zu diesem Zweck ist er mit Rucksack-Jet und Ungeziefer-Laser-Knarre bewaffnet und muß sich gegen zahlreiche Insekten zur Wehr setzen. Bei Berührung verliert er wertvolle Lebensenergie, die er allerdings wieder regenerieren kann, wenn er sich hinter Pflanzen versteckt. Bei »Tycoon Tex« muß der Spieler eine Ölpipeline bauen, während »Petals of Doom« ein Labyrinthspiel mit einem Hauch von »Pacman« ist. Alles in allem ist »C 16 Classics« eine tolle Sammlung von Spielen für alle Action-Fans.

Wer es besonders preiswert mag, der sollte einen Blick auf die Programme von Mastertronic werfen, die durch die Bank nur 10 Mark kosten. Am besten gefallen hat uns dabei »Kikstart«, ein pfiffiges Motorrad-Cross-Rennen für einen Spieler.

Als letztes stellen wir noch drei Spiele auf einer Kassette von CRL vor: Die Berks kommen in »The Berks-Trilogy« (Bild 4)! Berks sind niedlich aussehende, aber ungemein gefährliche Geschöpfe, die in drei verschiedenen Spielen (»Berks«, »Major Blink: Berks II« und »Berks III«) die Erde erobern wollen, was der Spieler natürlich verhindern muß. Die drei Spiele sind zwar recht ähnlich, aber trotzdem empfehlenswert, da sehr witzig gemacht. Außerdem ist es wirklich nicht einfach, auch nur eines der drei Spiele zu schaffen. Inzwischen gibt es noch einen vierten Teil, der den drei ersten aber dann doch zu sehr ähnelt und den man sich nicht extra zu der Sammlung der ersten drei kaufen muß.

Damit haben wir unseren kleinen Rundgang durch die C 16-Spielewelt beendet. Wir hoffen, daß Sie die eine oder andere Anregung für den nächsten verregneten Nachmittag gefunden haben. Und vielleicht finden Sie ja an den Computerspielen Geschmack und kaufen sich einen anderen Computer, für den das Spieleangebot größer und die Spiele qualitativ noch besser sind. (bs)

Info: Kingsoft, Schnackebusch 4, 5106 Roetgen
Mastertronic, Kaiser-Otto-Weg 18, 4770 Soest
Rushware, An der Gümpgesbrücke 24, 4044 Kaarst 2
Die Spiele haben unterschiedliche Preise und kosten zwischen 10 und 40 Mark auf Kassette.

Ergänzen Sie jetzt Ihre 64'er -Sammlung

Schaffen Sie sich ein interessantes Nachschlagewerk und gleichzeitig e wertvolles Archiv!

Kennen Sie alle Ausgaben von 64'er? Suchen Sie einen ganz bestimmten Testbericht? Oder haben Sie einen Teil eines interessanten Kurses versäumt? Suchen Sie nach einer speziellen Anwendung?

Damit Sie jetzt fehlende Hefte mit »Ihrem« Artikel nachbestellen können, finden Sie auf diesen Seiten eine Zusammenstellung aller wesentlichen Artikel der Ausgaben 01 bis 12/85.

Und so kommen Sie schnell an die noch lieferbaren Ausgaben: Prüfen Sie, welche Ausgabe in Ihrer Sammlung noch fehlt, oder welches Thema Sie interessiert. Tragen Sie die Nummer dieser Ausgabe und das Erscheinungsjahr (z.B. 2/85) auf dem Bestellabschnitt der hier eingehefteten Bestell-Zahlkarte ein. Die ausgefüllte Zahlkarte einfach heraustrennen und Rechnungsbetrag beim nächsten Postamt einzahlen. Ihre Bestellung wird nach Zahlungseingang umgehend zur Auslieferung gebracht.

Aktuell

Stichwort	Titel	Seite	Ausgabe
Allgemeines	Commodore Gestern Heute Morgen	10	01/85
Computer	Amiga — Der neue Supercomputer	8	09/85
Interview	Interview mit David Crane (Game Designer)	146	06/85
Lernen	Schule braucht Computer (VAM-Computer)	9	06/85
Messen	International Chaos Communication Congress	15	03/85
	Heiße Messe in der Wüste: CES	8	03/85
	Hannover-Messe '85	8	06/85
	Hannover-Messe '85	8	07/85
	Chicago im Zeichen der CES	8	08/85
	Aktuelles von der C'85 in Köln	15	08/85
	Btx Total (Internationale Funkausstellung)	8	10/85
	PCW-Computermesse in London	8	11/85
	Neues von der Commodore-Fachausstellung 1985	8	12/85
Recht	Die neue Abmahnmasche — Vorsicht bei Programmangeboten	8	05/85
	Die Ex-Knacker — wo sind sie geblieben?	27	08/85
	Interview mit Raubkopierern (Section 8)	28	08/85
	Schützer kontra Knacki's	23	08/85
	Raub-Talkshow	12	08/85
	Das Urheberrechtsgesetz und Gedanken zu seiner Anwendung	21	08/85
	Änderung des Urheberrechtsgesetzes	162	09/85

Buchbesprechungen

Stichwort	Titel	Seite	Ausgabe
Anfänger	Goldmann Computer Compact	87	03/85
	Basic-Wegweiser für den C 64	86	05/85
	Alles über den C 64, Sachbuchreihe, Band 1	115	06/85
	Lehrspielzeug Computer: C 64/VC 20	112	11/85
	C 64 Computerhandbuch	171	11/85
	Einführungskurs: Commodore 64	144	12/85
Anwendung	Dienstprogramme VC 20, C 64 und SX	86	05/85
	Spaß an Mathe mit dem Commodore 64	88	07/85
	Mathe für die Oberstufe mit dem C 64	88	07/85
	Mathematische Routinen VC 20, Elektrotechnik/Elektronik	112	11/85
	Commodore 64-Listings, Band 2: Dateiverwaltung, Schule, Hobby	112	11/85
	Das Trainingsbuch zum Datamat	144	12/85
C 128	Bücher zum C 128	22	10/85
DFÜ	Das Mailbox-Jahrbuch: Nutz die Netze	112	11/85
Grafik	Grafik auf dem Commodore 64 (+Fehlert. 9/85)	86	05/85
	Einführung in CAD mit dem Commodore 64	128	06/85
	Grafik & Musik auf dem Commodore 64	88	07/85
	Verschiedene Grafikbücher zum C 64	115	08/85
Programmieren	Von Basic zu Assembler: Das Commodore-Buch, Band 4	115	06/85
	64 Intern	115	06/85
	Das Interface Age System-Handbuch zum C 64	115	06/85
	Das C 64 Buch, Band 5: Simons Basic Leitfaden	144	12/85
	Basicode	144	12/85
	Noch mehr Tips und Tricks zum 64er	144	12/85
Speichern	Das Kassettenbuch zum C 64 und VC 20	87	03/85
	Die Floppy 1541 (M&T)	88	07/85
Spiele	Rombachs C 64 Spielführer	87	03/85
	Commodore 64-Listings, Band 1, Spiele	112	11/85
	35 ausgesuchte Spiele für Ihren Commodore 64	171	1/85

64'er Extra

Stichwort	Titel	Seite	Ausgabe
Prozessor	Befehlssatz des 6502/6510-Prozessors	84	09/85
Grafik	Die Videochip-Register des C 64	92	10/85
Sound	Der SID-Chip, seine Register und Programmierung	92	11/85
Speicher	Die Speicherbelegung des C 64	96	12/85

Abenteuerlösungen

Stichwort	Titel	Seite	Ausgabe
Lösungen	Dallas-Quest Lösung	90	01/85
	Guncho Krill-Enchanter ist gelöst	44	03/85
	Infocom-Geheimnisse gelüftet?	49	05/85
	Des Rätsels Lösung: Amazon	149	06/85
	Activision-Adventures entschleiert (Mindshadow, Tracer Sanction)	36	12/85
	Eureka! — ich hab's!	37	12/85
	Lösungen zu Hitchhiker's Guide und Sorcerer	39	12/85

Spiele-Tests

Stichwort	Titel	Seite	Ausgabe
007	James Bond — A View to a Kill	156	09/85
Abenteuer	Abenteuerpaket 1	48	08/85
	Shadowfire	146	09/85
	The Quest — mit C 64 auf Suche nach Drachen	47	01/85
Action	Hexenküche	50	07/85
	Master of the Lamps	48	07/85
	Rescue on Fractalus	158	10/85
	Stellar 7	49	08/85
Construction Set	Mail Order Monsters	49	08/85
	Racing Destruction Set	60	08/85
Geschicklichkeit	Australopedicus Robustus	60	08/85
	Boulder Dash II	159	10/85
	Crystal Castles	60	07/85
	Gribbly's Day out	148	09/85
	Rock'n Bolt	48	08/85
	Thing on a Spring	159	10/85
	Tom + Zaga	48	01/85
Pseudo-Adventures	Roland's Rat Race	49	08/85
	Fourth Protocol und Frankie g.t.H.	162	11/85
Renner	Die Renner 1985: Meistverkaufte Spiele	34	12/85
Schach	Viermal Schachmatt: Verschiedene Schachprogramme	32	12/85
Simulation	Elite	148	09/85
	Jump Jet	148	09/85
	Super Huey Hubschraubersimulator	49	07/85
Sport	Boxspiele: Frank Bruno's B. + Barry McGuigan Champions. B.	49	12/85
	Handkantenschlag per Joystick: Karateka + Exploding Fist	165	11/85
	Nick Faldo Plays the Open (Golf)	159	10/85
	Rallye Speedway	49	07/85
	Slapshot (Eishockey)	60	07/85
	Summer Games II	146	09/85
	World Series Baseball	49	07/85
Diverses	New York City und Air Support	145	06/85

Hardware-Tips und Bauanleitungen

Stichwort	Titel	Seite	Ausgabe
Audio/Video	Mit 5 Mark zu neuen Dimensionen (Stereoanlage am C 64)	34	05/85
	Ein Monitor ist genug (RGB+Composite an C 128)	16	10/85
C 16	Alte Datasette am C 16	31	04/85
	Alter Joystick am C 16	35	05/85
Eingabegeräte	Der Hexer — Zusatztastatur für den MSE	48	10/85
EPROM	EPROMs im Expansion-Port	46	10/85
	EPROM-Trans — Die Super-Erweiterung	42	10/85
	Das 64'er EPROM-Programmiergerät, Teil 1	44	12/85
Floppy/Datasette	Diskettenlaufwerk 1541 selbst justiert	32	10/85
	Die Datasette streikt nie wieder (Anpassung des Tonkopfs)	34	10/85
IEC-Bus	Auf zu neuen Welten: IEC-Bus im Selbstbau (+Fehlerteufel 10/85)	44	07/85
Joystick	Joystick im Selbstbau	33	03/85
	Dauerfeuer-Adapter	46	08/85
RS232/V.24	Das 30-Mark-Interface (Selbstbau RS232)	29	03/85
	Genau betrachtet: Die RS232/V.24-Schnittstelle	80	05/85
Diverses	Userport-Display	36	05/85
	Reset-Taster für alle Fälle (+Fehlert. 9/85)	130	06/85
	Aus eins mach vier (absturzfreie Betriebssystemumschaltung)	41	07/85

Hardware-Grundlagen

Stichwort	Titel	Seite	Ausgabe
Computer	Was bringt der C 128?	26	11/85
Drucker	Welcher Drucker ist der Richtige? (Grundlagen)	15	05/85
	Hammerwerke — wie funktionieren Typenraddrucker	32	06/85
	Die Alternativen: Thermo-, Tintenstrahldrucker + Plotter	24	07/85
Eingabegeräte	Versteht Sie Ihr Computer? (Wie funktionieren Eingabegeräte)	44	09/85
Floppy	Floppy oder Datasette?	129	06/85
Monitore	Wie funktionieren sie, was ist beim Kauf zu beachten?	16	12/85
	Das Kabel zum Monitor: Welche Normen gibt es?	28	12/85
Peripherie	Grafikeingabegerät: Wie funktionieren sie?	30	08/85

Hardware-Tests

Stichwort	Titel	Seite	Ausgabe
Computer	Generationswechsel: Test C 16	16	01/85
	Erster ausführlicher Test C 128 PC (Teil 1)	16	06/85
	Erster ausführlicher Test C 128, PC (Teil 2)	17	07/85
DFÜ	Marktübersicht Modems & Akustikkoppler	32	07/85
Drucker	Vergleich: Drucker unter 700 Mark (Tests und Marktübersicht)	18	05/85
	Tests und Marktübersicht Typenraddrucker	35	06/85
	Test: Brother EP 44	27	07/85
	Brother TC-600	118	08/85
	Riteman C+	133	09/85
	Panasonic KX-P1091	134	09/85
	Star SG 10C	132	09/85
	Melchers CP-80X — wie hätten Sie's denn gern?	25	10/85
	Geheimtip: Der RFI DP 165	24	10/85
	Epson GX 80 — einer für alle	26	10/85
	MPS 803 — ein Drucker für alle Gelegenheiten?	40	1/85
	Epson JX-80 das vielfarbige Druck-Genie	38	11/85
	Epson FX-85 neue Referenz	42	11/85
	SP 1000 VC — Superstar mit Haken	41	11/85
	Der NEC-P2 — das fernöstliche Wunder	159	12/85
	DMPG9 — eine solide Sache	162	12/85
	Das Doppelleben des Joystick-Ports: 10er-Tastaturen	50	09/85
	Joysticks: Test und Marktübersicht (+Fehlerteufel 12/85)	19	11/85
	Es geht auch anders: Lightpens und Trackballs	22	11/85
EPROMer	Frisch gebrannt ist halb gespeichert (EPROM-Programmiergeräte im Test)	39	07/85
	QuickByte II — das Kraftpaket	14	10/85
Floppy/Datasette	Turbo-Floppies, zweite Generation: Speeddos plus + Prologic DOS	28	10/85
	Das große Rennen: Schnelle Bandlaufwerke	37	10/85
	Professionelle Floppylaufwerke für den C 64 (IEC-Floppies)	30	10/85
	Gut gekauft ist halb gespeichert (Marktübersicht Disketten)	38	10/85
Grafik	Die Videowerkstatt (Digitizer-Test)	32	05/85
	Digitalbilder m.d. C 64: PrintTechnik Digitizer	24	01/85
Interface	Hardware-Interface ganz weich: Test EC 64	23	01/85
	Gute Connections — Übersicht Schnittstellen	21	03/85
	Card/Print +6 — Das Allround-Interface	20	03/85
	Das Wi[illegible]emann-Centronics-Interface	18	03/85

Stichwort	Titel	Seite
	Erst ein IEC-Bus öffnet Tür und Tor (+Fehlert.4/6-85)	24
Monitore	Marktübersicht: Monochrome Monitore	30
Musik	Trommelwirbel: Test Digital Drums	45
	Die Musikhardware zum C 64	17
Roboter	Roboter selbst gebaut (Fischertechnik)	167
Scanner	So lernt Ihr Drucker lesen	30
Speicher	Speichertuning VC 20: Test 64 KByte Karte	26
Steuern	Flottes Türmchen: MEA-Interface	116

Kurse

Stichwort	Titel	Seite
Assembler	Assembler ist keine Alchimie, Teil 5	142
	Assembler ist keine Alchimie, Teil 7	124
	Assembler ist keine Alchimie, Teil 9	138
	Assembler ist keine Alchimie, Teil 10	127
	Assembler ist keine Alchimie, Teil 11	126
	Assembler ist keine Alchimie, Teil 12	109
	Assembler ist keine Alchimie, Teil 13 (Schluß)	143
C 128	Entdeckungsreise duch den C 128	42
Effektives Programieren	Müllabfuhr im Computer: Garbage Collection, Teil 1	122
	Finden mit System, eine neuartige Suchmethode, Teil 3	148
	Sortieren mit dem Computer, Teil 2	159
	Sortieren mit dem Computer, Teil 3	124
	Sortieren mit dem Computer, Teil 4	138
	Sortieren mit dem Computer, Teil 5	124
	Sortieren mit dem Computer, Teil 6 (Schluß)	150
Extern	C 64 extern — Der Weg nach draußen, Teil 1	144
	C 64 extern — Der Weg nach draußen, Teil 2	122
	C 64 extern — Der Weg nach draußen, Teil 3 (Schluß)	129
Floppy	In die Geheimnisse der Floppy eingetaucht, Teil 4	148
	In die Geheimnise der Floppy eingetaucht, Teil 5	130
	In die Geheimnise der Floppy eingetaucht, Teil 6	145
	In die Geheimnisse der Floppy eingetaucht, Teil 7 (Schluß)	116
	Directory-Manipulationen I	140
Floppy	Directory-Manipulationen II	163
Grafik	Hires 3 — 15 neue Basic-Befehle, Teil 2	136
	Hires 3 — Grafikkurs-Anwendung, Teil 3 (Schluß)	152
	Sprites ohne Geheimnisse	40
	Streifzüge durch die Grafikwelt, Teil 1	106
	Streifzüge durch die Grafikwelt, Teil 2	149
Logeleien	Logeleien, Teil 1	143
	Logeleien, Teil 2	136
	Logeleien, Teil 3 (Schluß)	115
Musik	Dem Klang auf der Spur, Teil 2	136
	Dem Klang auf der Spur, Teil 4	131
	Dem Klang auf der Spur, Teil 5	152
	Dem Klang auf der Spur, Teil 7	132
	Dem Klang auf der Spur, Teil 8	133
	Dem Klang auf der Spur, Teil 9	126
	Dem Klang auf der Spur, Teil 10 (Schluß)	157
Speicher	Memory Map mit Wandervorschlägen, Teil 3	126
	Memory Map mit Wandervorschlägen, Teil 5	144
	Memory Map mit Wandervorschlägen, Teil 7	120
	Memory Map mit Wandervorschlägen, Teil 8	140
	Memory Map mit Wandervorschlägen, Teil 9	129
	Memory Map mit Wandervorschlägen, Teil 10	112
	Memory Map mit Wandervorschlägen, Teil 11	133
	Memory Map mit Wandervorschlägen, Teil 12	146
	Memory Map mit Wandervorschlägen, Teil 13	146
Sprachen	Basic ist out — es lebe Forth	43
VC 20	Der gläserne VC 20, Teil 4	130
	Der gläserne VC 20, Teil 6 (Schluß)	155

Software-Tips

Stichwort	Titel	Seite
C 128	Erste Fragen und Antworten zum C 128	14
	Fragen und Antworten zum 128er	20
	Fragen und Antworten zum 128er	40
Drucker	Der MPS 802 lernt Deutsch	30
	Centronics-Interface für jeden Bedarf	78
Textverarbeitung	Software Corner — professionelle Programme richtig eingesetzt (Vizawrite-Tips)	174
Tips & Tricks	Autoboot beim C 64	66
	Verbindungsfreundlich (Parallelschnittstelle des VC 20)	91
	Undefinierte Opcodes des 6502	84
	Durch POKEs zum Erfolg (Spiele-POKEs)	83
	Tips und Erweiterungen zu Hi-Eddi und Simons Basic	88
	Basic-Befehle im Griff	79
	Durch POKEs zum Erfolg: Spiele-POKEs	76
	Formatierte Eingabe	145
	Hi-Text (Text in Hires)	70
	Verbotene Variablen	66
	Verschiedene Routinen für Anfänger und Profis (+Fehlerteufel 12/85)	85
	Der Trick mit dem Joystick (Joystickabfrage)	24
	Verschiedene Tips für Anfänger und Fortgeschrittene	106

Software-Grundlagen

Stichwort	Titel	Seite
Assembler	Assembler? Assembler! (Einführung)	32
	Assembler-Bedienung leicht gemacht, Teil 1	169
DFÜ	Der erste Kontakt mit DFÜ	42
	Die Netze der Post: Btx, Datex-P, Telebox	45
	DFÜ — Was ist das?	44
	Mailbox für Anfänger	30

...gs zum Abtippen

...ware-Tests

So machen's andere

Die Ausgaben 2/85 und 4/85 sind bereits vergriffen und nicht mehr lieferbar!

Am besten gleich mitbestellen: Die praktischen 64'er-Sammelboxen



Ein kompletter Jahrgang (12 Ausgaben) paßt in eine der praktischen Sammelboxen! Am besten gleich mitbestellen!

Für alle Leser, die »64'er« regelmäßig kaufen, sammeln oder im Abonnement beziehen, gibt es jetzt ein interessantes Service-Angebot: die 64'er-Sammelbox!

Mit dieser Sammelbox bringen Sie nicht nur Ordnung in Ihre wertvollen Hefte, sondern schaffen sich gleichzeitig ein interessantes und attraktives Nachschlagewerk.

Übrigens: Die Sammelbox ist nicht nur ein praktisches Aufbewahrungsmittel: Sie eignet sich auch hervorragend als Geschenk für Freunde und Bekannte zu vielen Anlässen.

Auch die bisher erschienenen Sonderhefte können Sie jetzt direkt bestellen:

SONDERHEFT 01/84: TIPS & TRICKS
Unentbehrliche Anwendungslistings für C 64 und VC 20.

SONDERHEFT 02/85: ABENTEUERSPIELE 1
Fesselnde Adventures mit zahlreichen Lösungen und einem Programmierkurs.

SONDERHEFT 03/85: SPIELE
Heiße Listings für Spiele-Fans und eine große Marktübersicht.

SONDERHEFT 04/85: GRAFIK & DRUCKER
Von der 3D-Darstellung bis zur Hardcopy-Routine.

SONDERHEFT 05/85: FLOPPY/DATASETTE
Soft-Tools zum komfortablen und noch schnelleren Betrieb von Floppy und Datasette.

SONDERHEFT 06/85: AUSGEWÄHLTE SUPER-LISTINGS
Top-Themen aus 64'er bringt eine Auswahl der besten 64'er Programme.

SONDERHEFT 07/85: ANWENDUNGEN/DFÜ
Leistungsfähige Programme für professionelle Anwendungen und Datenfernübertragung.

SONDERHEFT 08/86: ASSEMBLER
Assembler-Know-how für Anfänger und Fortgeschrittene.

SONDERHEFT 01/86: PC 128
Komplette Beschreibungen von C 128 und C 128D und passendem Zubehör. Die Unterschiede zum C 64.

SONDERHEFT 02/86: TIPS & TRICKS
Super-Listings, ausführliche Grundlagen und die besten Tips&Tricks und Einzeiler aus 64'er.

SONDERHEFT 03/86: C16, C116, VC20 UND PLUS 4
Umfassende Grundlagen und aktuelle Informationen zu C 16, C 116, VC 20 und Plus 4.

SONDERHEFT 04/86: ABENTEUERSPIELE 2
Auf 160 Seiten alles über das Programmieren von Abenteuerspielen und Super-Listings zum Abtippen.

SONDERHEFT 05/86: C64-GRUNDWISSEN
Für alle Einsteiger umfassende Grundlagen und Hilfestellungen rund um den C 64.

SONDERHEFT 06/86: GRAFIK
Grafikprogrammierung des C 64, C 128 und C 128 im C 64-Modus. Dreidimensional konstruieren mit »Giga-CAD«.

SONDERHEFT 07/86: PEEKs UND POKEs
Einführungskurs in die wichtigsten Speicherstellen für C 64, C 16 und C 128. Über 30 Seiten Tips & Tricks.

Tragen Sie die Nummer des gewünschten Sonderheftes (z.B. 09/85) auf dem Bestellabschnitt der hier eingehefteten Bestell-Zahlkarte ein.

Videofilme im Griff

Wer etwas Ordnung in seine Videofilm-Sammlung bringen möchte, kann auch den Computer sinnvoll einsetzen.

In vielen Haushalten steht heutzutage ein Videorecorder, und so gibt es inzwischen eine ganze Reihe von Sammlern, die sich die interessantesten Spielfilme aufzeichnen und aufbewahren. Um nun in diese Sammlung etwas Ordnung zu bringen, verwendet man im Zeitalter des Computers natürlich seinen C 16. Mit dem folgenden Programm (Listing) können Sie Ihre Videofilme eingeben und speichern und natürlich jederzeit auch wieder laden und ändern. Sie können nach bestimmten Begriffen suchen und eine komplette Liste auf einem Drucker ausgeben. Das Speichern der Videodatei kann nur auf Datasette erfolgen, für Diskettenbetrieb müßte das Programm extra angepaßt werden. Die Bedienung des Programms ist relativ einfach und wird auch vom Programm selber erklärt.

Dateiverwaltung

Das vorliegende Programm ist ein Beispiel für eine Dateiverwaltung. Eine Ansammlung von Daten wird eingegeben und gespeichert und kann bei Bedarf wieder geladen und durchgesehen werden. Vergleichbar ist das in etwa mit einem Karteikasten. Der Unterschied liegt in der Verarbeitungsgeschwindigkeit: Wie lange würden Sie benötigen, um einhundert Karteikarten zu sortieren? Der Computer benötigt dafür meist nur Sekunden!

Leider hat der C 64 nur eine begrenzte Speicherkapazität, deswegen kann man nur geringe Datenmengen speichern. Immerhin erhalten Sie so aber einen Einblick in die Dateiverwaltung und können die so gewonnenen Erkenntnisse auch verwenden, wenn Sie an einem größeren Computer arbeiten werden.

(Dieter Beer/bs)

```
1 REM  ***********************
2 REM  *                     *
3 REM  *  COMMODORE 16/116   *
4 REM  *                     *
5 REM  *    DIETER BEER      *
6 REM  *                     *
7 REM  ***********************
10 REM
12 PRINT CHR$(27) CHR$(78)
13 KEY 1,"ACTION": KEY 2,"WESTERN": KEY 3,"SPASS
   ": KEY 8,"SCIENCE-FICTION": KEY 4,"HORROR"
14 KEY 5,"MUSIK"
15 COLOR 0,7
20 DIM D$(87,7)
25 GOSUB 50000
30 F$(1)=" TITEL{7SPACE}"
31 F$(2)=" FILMART{5SPACE}"
32 F$(3)=" CASS.-NR.{3SPACE}"
33 F$(4)=" SEITE{7SPACE}"
34 F$(5)=" STARTZEIT{3SPACE}"
35 F$(6)=" SPIELDAUER{2SPACE}"
36 F$(7)=" BEWERTUNG{3SPACE}"
99 GOTO 1000
100 REM   PROGRAMMKOPF
130 PRINT "{CLR}"
140 FOR I=1 TO 40: PRINT "=";: NEXT
150 PRINT "{12SPACE}VIDEOVERWALTUNG"
160 FOR I=1 TO 40: PRINT "=";: NEXT
170 PRINT
180 RETURN
200 REM    FEHLERMELDUNG
230 PRINT CHR$(19);
240 FOR X=1 TO 21: PRINT CHR$(17);: NEXT
250 PRINT CHR$(18);FE$;CHR$(146);
260 FOR X=1 TO 1000: NEXT
270 RETURN
300 REM    KOEPFE PROGRAMMTEILE
330 PT$(1)="{3SPACE}DATEI LADEN{3SPACE}"
340 PT$(2)=" DATEI SPEICHERN "
350 PT$(3)=" FILME{3SPACE}EINGEBEN"
360 PT$(4)=" FILME{3SPACE}AENDERN "
370 PT$(5)=" FILME{3SPACE}LOESCHEN"
380 PT$(6)=" FILME{3SPACE}AUSGEBEN"
385 PT$(7)=" TITEL{3SPACE}LISTING "
390 PT$(8)="{4SPACE}ERKLAERUNG{3SPACE}"
395 PT$(9)="PROGRAMM BEENDEN "
400 PRINT "{CLR}"
410 PRINT CHR$(30);
420 PRINT "UCCCCCCCCCCCCCCCCCI"
430 PRINT "B";PT$(F);"B"
440 PRINT "JCCCCCCCCCCCCCCCCCK"
450 RETURN
500 REM    DATEN IM RECHNER
530 IF Z>0 THEN 560
540 FE$="KEINE DATEN IM RECHNER"
550 GOSUB 200
560 RETURN
1000 VOL 8: SOUND 1,900,5: GOSUB 100
1005 PRINT CHR$(30);"UCCCCCCCCCCCCCCCCCCCI"
1010 PRINT "B";"PROGRAMMFUNKTIONEN:";"B"
1015 PRINT "JCCCCCCCCCCCCCCCCCCCK"
1030 PRINT
1035 PRINT " {RVSON,27SPACE}"
1040 PRINT " {RVSON,3SPACE}-1-{4SPACE}DATEI LADE
     N{6SPACE}"
1050 PRINT " {RVSON,3SPACE}-2-{4SPACE}DATEI SPEI
     CHERN{2SPACE}"
1060 PRINT " {RVSON,3SPACE}-3-{4SPACE}FILME EING
     EBEN{3SPACE}"
1070 PRINT " {RVSON,3SPACE}-4-{4SPACE}FILME AEND
     ERN{4SPACE}"
1080 PRINT " {RVSON,3SPACE}-5-{4SPACE}FILME LOES
     CHEN{3SPACE}"
1090 PRINT " {RVSON,3SPACE}-6-{4SPACE}FILME AUSG
     EBEN{3SPACE}"
1100 PRINT " {RVSON,3SPACE}-7-{4SPACE}TITEL LIST
     ING{4SPACE}"
1101 PRINT " {RVSON,3SPACE}-8-{4SPACE}ERKLAERUNG
     {7SPACE}"
1102 PRINT " {RVSON,3SPACE}-9-{4SPACE}PROGRAMM B
     EENDEN "
1103 PRINT " {RVSON,27SPACE}"
1105 PRINT
1110 PRINT "{5SPACE}AUSWAHL "
1115 INPUT "{6SPACE}";F
1120 IF F=0 OR F>9 THEN FE$="UNGUELTIGER WERT":
     GOSUB 200: GOTO 1000
1125 VOL 8: SOUND 1,900,5
1130 ON F GOTO 5000,10000,15000,20000,25000,3000
     0,40000,45000,35000
5000 REM   DATEI LADEN
5030 GOSUB 300: PRINT
5040 PRINT "BITTE DATENKASSETTE EINLEGEN{4SPACE}
     "
5050 PRINT "{DOWN}UND ZURUECKSPULEN !"
5060 PRINT "{4DOWN}DRUECKE DANACH RETURN "
5070 INPUT X$
5080 PRINT
5090 OPEN 1,1,0,"VIDEO"
5100 INPUT#1,Z
5110 FOR Y=1 TO Z
5120 FOR I=1 TO 7
5130 INPUT#1,D$(Y,I)
5140 NEXT I
5150 NEXT Y
5160 CLOSE 1
5170 PRINT "DATEN SIND GELADEN !"
5180 FOR X=1 TO 2000: NEXT
5190 GOTO 1000
10000 REM   DATEI SPEICHERN
10040 GOSUB 300: PRINT
10050 GOSUB 500: IF Z=0 THEN 1000
10060 PRINT "BITTE DATENKASSETTE EINLEGEN "
```

```
10070 PRINT "{DOWN}UND ZURUECKSPULEN !": PRINT
10080 PRINT "{DOWN}DRUECKE DANACH{3SPACE}RETURN"
10090 INPUT X$
10100 OPEN 1,1,1,"VIDEO"
10110 PRINT#1,Z
10120 FOR Y=1 TO Z
10130 FOR I=1 TO 7
10140 PRINT#1,D$(Y,I)
10150 NEXT I
10160 NEXT Y
10170 CLOSE 1
10180 PRINT : PRINT "DATEN SIND GESPEICHERT !"
10190 FOR X=1 TO 2000: NEXT
10200 GOTO 1000
15000 REM   FILME EINGEBEN
15030 GOSUB 300
15040 Z=Z+1
15050 PRINT
15060 FOR I=1 TO 7
15070 PRINT F$(I);
15080 INPUT D$(Z,I)
15090 PRINT : NEXT
15100 PRINT
15110 PRINT "EINGABE RICHTIG (J/N)"
15120 X$="": INPUT X$
15130 IF X$="J" THEN 15160
15140 IF X$="N" THEN Z=Z-1: GOTO 15000
15150 PRINT CHR$(145);: GOTO 15110
15160 PRINT "WEITERE EINGABEN (J/N)"
15170 X$="": INPUT X$
15180 IF X$="J" THEN 15000
15190 IF X$="N" THEN 1000
15200 PRINT CHR$(145);: GOTO 15160
20000 REM     FILME AENDERN
20030 ZZ=1
20040 GOSUB 300
20050 PRINT
20060 FOR I=1 TO 7
20070 PRINT I;F$(I);D$(ZZ,I): PRINT
20080 NEXT
20090 GET X$: IF X$="" THEN 20090
20100 IF X$<>"{LEFT}" AND X$<>"{RIGHT}" AND X$<>
      " " AND X$<>"{UP}" THEN 20090
20110 IF X$="{UP}" THEN 1000
20120 IF X$="{RIGHT}" AND ZZ<Z THEN ZZ=ZZ+1: GOT
      O 20040
20130 IF X$="{LEFT}" AND ZZ>1 THEN ZZ=ZZ-1: GOTO
       20040
20140 IF X$=" " THEN 20160
20150 GOTO 20090
20160 PRINT
20170 INPUT "FELDNUMMER (1-7) ";X
20180 IF X<1 OR X>7 THEN PRINT CHR$(145);GOTO 20
      170
20190 PRINT
20200 INPUT "NEUER INHALT : ";D$(ZZ,X)
20210 GOTO 20040
25000 REM    FILME LOESCHEN
25030 GOSUB 300
25040 PRINT
25050 GOSUB 500
25060 IF Z=0 THEN 1000
25065 INPUT "TITEL{2SPACE}: ";D1$
25080 X=1
25090 IF D$(X,1)=D1$ THEN 25135
25100 IF X<Z THEN X=X+1: GOTO 25090
25110 FE$=" NAME NICHT GEFUNDEN !"
25120 GOSUB 200: GOTO 1000
25130 GOSUB 300: PRINT
25135 PRINT "{2UP}"
25140 FOR I=1 TO 7
25145 PRINT F$(I);D$(X,I): PRINT
25150 NEXT : PRINT
25160 PRINT "FILM{2SPACE}LOESCHEN ? (J/N) ": INP
      UT X$
25170 IF X$="J" THEN 25200
25180 IF X$="N" THEN 1000
25190 PRINT CHR$(145);: GOTO 25160
25200 FOR Y=X TO Z-1
25210 FOR I=1 TO 7
25220 D$(Y,I)=D$(Y+1,I)
25230 NEXT I
25240 NEXT Y
25250 Z=Z-1
25260 GOTO 1000
30000 REM     FILME AUSGEBEN
30030 GOSUB 300: PRINT
30040 GOSUB 500
30050 IF Z=0 THEN 1000
30060 PRINT "DRUCKER ODER BILDSCHIRM (D/B) ": IN
      PUT G$
30070 IF G$<>"B" AND B$<>"D" THEN PRINT CHR$(145
      );: GOTO 30060
30075 IF G$="D" THEN OPEN 1,4
30080 GOSUB 300: PRINT
30090 PRINT "SUCHBEGRIFFE:"
30100 PRINT "-------------"
30110 PRINT
30120 FOR I=1 TO 7: S$(I)="": NEXT
30130 FOR I=1 TO 7
30140 PRINT F$(I);
30150 INPUT S$(I)
30160 PRINT : NEXT
30170 FOR Y=1 TO Z
30180 S=0
30190 FOR I=1 TO 7
30200 IF S$(I)="" THEN S=S+1: GOTO 30220
30210 IF D$(Y,I)=S$(I) THEN S=S+1
30220 NEXT I
30230 IF S<>7 THEN 30300
30240 IF G$="D" THEN PRINT#
30245 IF G$="B" THEN GOSUB 300: PRINT
30250 FOR I=1 TO 7
30260 IF G$="B" THEN PRINT F$(I);D$(Y,I): PRINT
30270 IF G$="D" THEN PRINT#1,F$(I);D$(Y,I)
30280 NEXT I
30285 IF G$="D" THEN 30300
30290 PRINT : INPUT "DRUECKEN SIE{2SPACE}RETURN"
      ;X$
30300 NEXT Y
30310 GOSUB 300: PRINT : PRINT
30320 PRINT "DATEI ENDE "
30330 INPUT "DRUECKE{3SPACE}RETURN";X$
30340 IF G$="D" THEN CLOSE 1
30350 GOTO 1000
35000 REM PROGRAMM BEENDEN
35030 GOSUB 300: PRINT
35040 IF Z=0 THEN 35150
35050 PRINT "SIND ALLE DATEN GESICHERT (J/N)"
35060 INPUT X$
35070 IF X$="N" THEN 1000
35080 IF X$="J" THEN 35100
35090 PRINT CHR$(145);: GOTO 35050
35100 GOSUB 300
35110 PRINT "{DOWN}DAS PROGRAMM KANN MIT"
35120 PRINT "{DOWN,5SPACE}'GOTO 1000'"
35130 PRINT "{DOWN}WIEDERGESTARTET WERDEN, OHNE
      DASS"
35140 PRINT "{DOWN}DATEN VERLOHREN GEHEN"
35150 PRINT
35160 END
40000 PRINT "{CLR,DOWN,13SPACE}TITEL LISTING"
40010 PRINT "{12SPACE}###############{2DOWN}"
40020 FOR U=1 TO Z: PRINT " "D$(U,1);: PRINT TAB
      (25)"{RVSON}"D$(U,2)"{DOWN}"
40030 GET KEY T$
40040 NEXT U
40050 PRINT "{3DOWN,12RIGHT}DRUECKE >RETURN<"
40060 GET KEY O$
40070 IF O$<>CHR$(13) THEN 40060
40080 GOTO 1000
45000 PRINT "{CLR,2DOWN,14RIGHT}ERKLAERUNG"
45010 PRINT "{12RIGHT}##############"
45020 PRINT "{4DOWN,3RIGHT}DURCH DRUECKEN DER EN
      TSPRECHENDEN"
45030 PRINT "{DOWN,3RIGHT}FUNKTIONSTASTEN LASSEN
       SICH DIE"
45040 PRINT "{DOWN,3RIGHT}VERSCHIEDENEN FILMARTE
      N EINGEBEN!"
45050 PRINT "{6DOWN,7RIGHT,4SPACE}> EINE TASTE <
      "
45060 GET KEY PP$: GOTO 1000
50000 PRINT "{CLR,22DOWN}"
50005 PRINT "{RVSON} 1 - ACTION{3SPACE}2 - WESTE
      RN{3SPACE}3 - SPASS{3SPACE}"
50010 PRINT "{RVSON} 4 - HORROR{3SPACE}5 - MUSIK
      {5SPACE}8 - S.F.{3SPACE,HOME}"
50100 PRINT "{HOME}" CHR$(27) CHR$(84)
50110 PRINT "{HOME,23DOWN,LEFT}" CHR$(27) CHR$(6
      6): RETURN
```

Listing. Video-Verwaltung.
Beachten Sie bitte die Eingabehinweise auf Seite 130.

Schachdiagramme mit dem C 16

Plotten Sie Schachdiagramme auf dem VC 1520. Der schachbegeisterte Computer-Besitzer erhält damit die Möglichkeit, Eröffnungsvarianten karteikartenähnlich zu sammeln oder sich eine Gedankenstütze fürs Fernschach zu erstellen.

Wie man auf den beiden Diagrammen leicht erkennen kann, sind sie mit zwei verschiedenen Figurensätzen erstellt worden. Eine einzelne Figur besteht aus zirka 45 Linien. Ein Figurensatz besteht also aus etwa 5480 (45 x 2 x 6) Daten. Da diese beiden Figurensätze eine reine DATA-Wüste im Listing verursachen würden und das Aussehen der Figuren ohnehin Geschmackssache ist, sind sie im Listing nicht zu finden. Sie sind mit dem »SATZERSTELLER« (Listing 2) erstellt worden. Mit Hilfe der Satzerstellers lassen sich Figurensätze in etwa 20 Minuten erstellen. Der Satzersteller bietet folgende Möglichkeiten:

- Mit Hilfe des Joysticks die Figur bequem und bildgetreu zu erstellen
- Ein Minicursor und eine Koordinatenanzeige geben die Cursorposition an
- Alle Punktkoordinaten werden auf dem Bildschirm ausgegeben
- auf dem Bildschirm wird jeder Schritt wie auf dem Plotter nachvollzogen
- Die Figur wird auf dem Plotter gezeichnet
- Editiermöglichkeiten: (Joystick in Port 1)
- Punkt setzen: FEUER
- Punkt löschen: UNTEN + FEUER
- Figur spiegeln: OBEN + FEUER
- Neuer Anfangspunkt: RECHTS + FEUER, dann Punkt setzen
- Figur plotten: Taste <P>

Diagramme wie gemalt

Bei der Figurenerstellung kann man folgendermaßen vorgehen:

1. Schritt (nur für Diskettenlaufwerk): ein sequentielles File eröffnen: OPEN2,8,2,"FILENAME,S,W":CLOSE2
In Zeile 480: 480 OPEN2,8,2,"FILENAME,S,A" entsprechend den Filenamen einsetzen

1 a) für Datasette: Zeile 480 in OPEN2,1,1 "FIGUREN" ändern

2. Die Figuren müssen in folgender Reihenfolge erstellt werden: Bauer - Springer - Läufer - Turm - Dame - König

3. Es ist günstig, für alle Figuren die gleiche Starthöhe (in den Diagrammen 44) zu benutzen.

4. Die Figur darf aus 56 Punkten bestehen. (Mit Rücksicht auf 2048 Byte freien Speicher konnten die Arrays in Zeile 10 nicht größer gewählt werden.)

5. Nach Druck der Taste <P> erscheint nach dem Plotten die Frage "DISK?" Bei Eingabe von <J> werden die Figurdaten an das Diskfile angehängt (Datasette: Daten als neues File). Bei <N> wird die Figur nicht gespeichert. Dann erscheint: ZURÜCK/NEU/ENDE. Bei <Z> geht man ins Bild mit der gerade erstellten Figur zurück. <N>: Neustart, sonst Programmende.

Als Beispiel die Erstellung von »Satz 2«:

Eine wichtige Hilfe zur Figurenerstellung ist die SPIEGELFUNKTION. Nach Wählen dieser wird an einer senkrechten gedachten Linie durch den letzten Punkt die Figur auf die

```
10 DIMBX%(60,5),CY%(60,5),BR%(7,7):OPEN7,6,2:OPEN5
,6,3:OPEN4,6:VOL8
20 PRINTCHR$(147)
30 FORT=0TO1
40 IFT=1THENPOKE2023,19:PRINT"{HOME}STELLUNG SCHWA
RZ :":ELSEPRINT"{HOME}STELLUNG WEISS :"
50 PRINT
60 FG=7
70 FG=FG-1:IFFG=0THEN190
80 READD$:SOUND1,600,20
90 PRINTD$" :{SPACE,RVSON,SPACE,RVOFF}";:POKE239,0
:SOUND1,700,20
100 GETKEYA$:KE=ASC(A$):IFA$="-"THENPRINT"{LEFT}"A
$:GOTO70
110 IFKE<65ORKE>72THEN100
120 PRINT"{LEFT}"A$"{RVSON,SPACE,RVOFF}";
130 GETKEYA$:KF=ASC(A$):IFKF=20THENPRINT"{LEFT,SPA
CE,2LEFT,RVSON,SPACE,RVOFF}";:GOTO100
140 IFKF<49ORKF>56THEN130
150 PRINT"{LEFT}"A$"{RVSON,SPACE,RVOFF}";
160 GETKEYA$:IFASC(A$)<>20ANDASC(A$)<>13THEN160
170 IFASC(A$)=20THENPRINT"{LEFT,SPACE,2LEFT,RVSON,
SPACE,RVOFF}";:GOTO130
180 BR%(KE-65,KF-49)=FG+T*10:PRINT"{LEFT }":GOTO90
190 RESTORE:NEXT:POKE2023,0
200 PRINT"{CLR}DIAGRAMM MIT TEXT? (J/N)":GETKEYG$:
IFG$="N"THEN370
210 PRINT"{CLR}STANDARDMUSTER ?(J/N)":GETKEYG$:IFG
$="N"THENPRINT"{CLR}":GOTO270
220 PRINT"{CLR}TURNIER :"
230 PRINT:PRINT"PARTIE NR.:"
240 PRINT"WEISS{3SPACE}:"
250 PRINT"SCHWARZ :"
260 PRINT:PRINT"STELLUNG NACH DEM{5SPACE}ZUG VON"
270 FORI=3432TO3471:POKEI,160:NEXT
280 OPEN14,0:PRINT"{HOME}"
290 INPUT#14,A$
300 PRINT#5,1:FORI=3072TO3431:BI=PEEK(I)
310 IFBI<32THENBI=BI+64
360 PRINT#4,CHR$(BI);:NEXT:PRINT#4,"":PRINT#4,""
370 PRINT"{CLR}WELCHEN FIGURENSATZ?":PRINT
380 RESTORE2000
390 TA=TA+1:READSF$(TA):IFSF$(TA)="E"THEN410
400 PRINT:PRINTTA". "SF$(TA):GOTO390
410 GETKEYG$:NJ=VAL(G$):IFNJ<1ORNJ>TA-1THEN410
420 PRINT:PRINT"DISKETTE/CASSETTE (D/C)"
430 GETKEYGE$:IFGE$<>"D"ANDGE$<>"C"THEN430
440 IFGE$="D"THENOPEN2,8,2,SF$(NJ)+",S,R":ELSEOPEN
2,1,0,SF$(NJ)
450 FORP=0TO5:INPUT#2,I(P):FORG=0TOI(P):INPUT#2,BX
%(G,P):INPUT#2,CY%(G,P):NEXT:NEXT
460 CLOSE2
470 OPEN1,6,1:PRINT#7,0
480 FORI=0TO8:PRINT#1,"M",94,-I*48:PRINT#1,"D",477
,-I*48:PRINT#1,"M",I*48+94,0
490 PRINT#1,"D",94+I*48,-384:NEXT
500 FORI=0TO7:Y=-I*48:FORG=0TO7:X=G*48+94
510 IF(I+G)/2=INT((I+G)/2)THEN540
520 FORD=0TO3:PRINT#1,"M",X,-D*12+Y:PRINT#1,"D",48
-D*12+X,-48+Y:NEXT
530 FORD=0TO2:PRINT#1,"M",D*12+X+12,0+Y:PRINT#1,"D
",48+X,-36+12*D+Y:NEXT
540 NEXT:NEXT
550 FORM=7TO0STEP-1:Y=(M-7)*48-48
560 FORN=0TO7:X=N*48+94
570 IFBR%(N,M)=0THEN650
580 IFBR%(N,M)>9THENP=BR%(N,M)-11:PRINT#7,1:GOTO60
0:ELSEPRINT#7,3
590 P=BR%(N,M)-1
600 G=0
610 PRINT#1,"M",BX%(G,P)+X,48-CY%(G,P)+Y
620 G=G+1:IFG=I(P)THEN650
630 IFBX%(G,P)=99THENG=G+1:GOTO610
640 PRINT#1,"D",BX%(G,P)+X,48-CY%(G,P)+Y:GOTO620
650 NEXT:NEXT
660 PRINT#1,"M",108,2
670 PRINT#7,0:PRINT#5,2:CLOSE5
680 PRINT#4,"A B C D E F G H";
690 FORM=0TO7:Y=-37-M*48:PRINT#1,"M",60,Y;
700 PRINT#4,CHR$(56-M);:NEXT:SYS65511
710 DATA"KOENIG","DAME","TURM","LAEUFER","SPRINGER
","BAUER"
2000 DATA"SATZ","2. SATZ","E"
```

Listing 1. »Diagrammdrucker«. Vom C 16 auf den VC 1520.

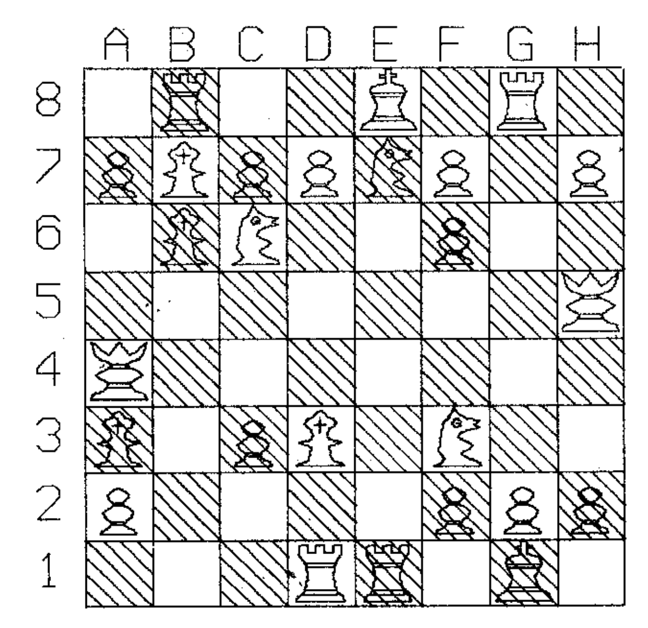
```
10 DIMBX%(55),CY%(55):VOL8:COLOR0,15,2:COLOR4,1:CO
LOR1,2:GRAPHIC1,1
20 X=160:Y=100:DRAW1,X,Y:BOX1,135,75,185,125
30 GOSUB60:LOCATEX+C,Y+D:IFRDOT(2)=1THEN30
40 DRAW0,X,Y:X=X+C:Y=Y+D:DRAW1,X,Y:A$=RIGHT$(STR$(
X-136),2):B$=RIGHT$(STR$(Y-76),2)
50 CHAR1,4,2,"X-WERT :":CHAR1,4,4,"Y-WERT :":CHAR1
,14,2,A$:CHAR1,14,4,B$:GOTO30
60 A=JOY(2):IFA=128ANDK=0THEN220
70 IFA=129THEN300
80 IFA=133ANDK=0ANDI>0THEN370
90 GETA$:IFA$="P"THEN420
100 IFA<>131ORK=1THEN120
110 K=1:BX%(I)=99:CY%(I)=99:SOUND1,550,20:O=1:GOSU
B260:GOTO60
120 IFA=0ORA>127THEN60
130 K=0:ONAGOTO140,150,160,170,180,190,200,210
140 C=0:D=-1:RETURN
150 C=1:D=-1:RETURN
160 C=1:D=0:RETURN
170 C=1:D=1:RETURN
180 C=0:D=1:RETURN
190 C=-1:D=1:RETURN
200 C=-1:D=0:RETURN
210 C=-1:D=-1:RETURN
220 SOUND1,700,20:BX%(I)=X-136:CY%(I)=Y-76:IFY>100
THENY=Y-2:ELSEY=Y+2
230 DRAW1,X,Y:K=1:IFI=0THEN260
240 IFBX%(I-1)=99THEN260
250 DRAW1,BX%(I-1)+136,CY%(I-1)+76TOBX%(I)+136,CY%
(I)+76
260 C$=RIGHT$(STR$(I+1),2)+". "+RIGHT$(STR$(BX%(I)
),2)+","+RIGHT$(STR$(CY%(I)),2)
270 IFI>15THENP=25:Q=-15:ELSEP=2:Q=7
280 IFI>39THENP=25:Q=-39
290 CHAR1,P,I+Q,C$:I=I+1:IFO=1THENO=0:RETURN:ELSEG
OTO60
300 L=I-1:J=BX%(L)
310 FORG=1TOI-1:IFBX%(L-G)<>99ANDTA<>1THEN340
315 IFTA=1THENBX%(I)=-BX%(L-G)+2*J:CY%(I)=CY%(L-G)
:TA=0:GOTO330
320 BX%(I)=BX%(L-G):CY%(I)=CY%(L-G):TA=1
330 O=1:GOSUB260:GOTO360
340 CY%(I)=CY%(L-G):BX%(I)=-BX%(L-G)+2*J
350 O=1:GOSUB250
360 NEXT:BX%(I)=BX%(0):CY%(I)=CY%(0):GOTO250
370 IFI=0THEN60
380 I=I-1:IFI>0ANDBX%(I)<>99THENDRAW0,BX%(I-1)+136
,CY%(I-1)+76TOBX%(I)+136,CY%(I)+76
390 IFI>15THENP=25:Q=-15:ELSEP=2:Q=7
400 SOUND1,600,20:K=1:IFI>39THENP=25:Q=-39
410 CHAR1,P,I+Q,"{11SPACE}":GOTO60
420 OPEN1,6,1:G=0
430 PRINT#1,"M",220+BX%(G),48-CY%(G)
440 G=G+1:IFG=ITHEN470
450 IFBX%(G)=99THENG=G+1:GOTO430
460 PRINT#1,"D",220+BX%(G),48-CY%(G):GOTO440
470 GRAPHIC0:PRINT"{CLR}DISK.":POKE239,0:GETKEYA$:
IFA$="N"THEN500
480 OPEN2,8,2,"FILENAMEN,S,A"
490 PRINT#2,I:FORG=0TOI:PRINT#2,BX%(G):PRINT#2,CY%
(G):NEXT:CLOSE2
500 PRINT"{CLR}ZURUECK/NEU/ENDE":POKE239,0:GETKEYA
$:IFA$="Z"THENSYS65511:GRAPHIC1,0:GOTO60
510 IFA$<>"E"THENRUN
```

Listing 2. »Satzersteller«, für Ihr eigenes Design

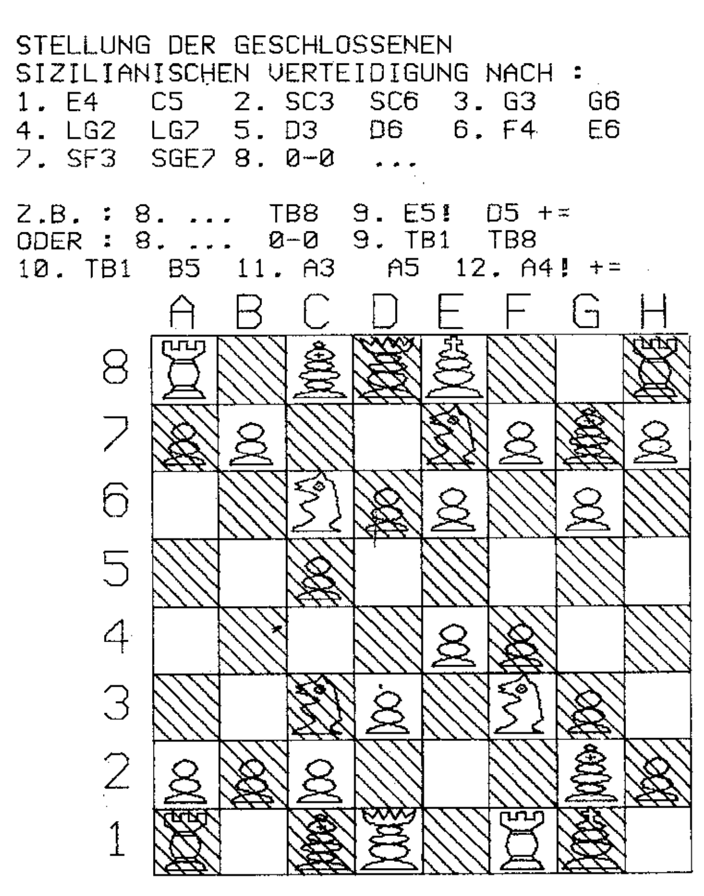
```
10 DIMI%(800)
20 PRINT"{CLR}DIE CASSETTE BIS ZUR ERSTEN FIGUR SP
ULEN"
30 PRINT:PRINT"DANN TASTE 'F' DRUECKEN"
40 GETKEYG$:IFG$<>"F"THEN40
50 FORJ=1TO6
60 OPEN2,1,0,"FIGUREN"
70 INPUT#2,I%(G)
80 TA=I%(G)
90 FORN=1TOTA:INPUT#2,I%(G+N):NEXT:G=G+TA+1
100 CLOSE2:NEXT
110 PRINT"{CLR}ZIELCASSETTE EINLEGEN UND 'Z' DRUEC
KEN"
120 GETKEYG$:IFG$<>"Z"THEN120
130 INPUT"SATZNAME";SA$
140 OPEN2,1,1,SA$
150 FORU=0TOG-1:PRINT#2,I%(U):NEXT:CLOSE2
160 PRINT:PRINT"FERTIG"
```

Listing 3. »Tape-Hilfe« ermöglicht die Verwendung einer Datasette

```
        SCHLUSSKOMBINATION DER
        'IMMERGRUENEN' PARTIE
     A.ANDERSSEN    -    J.DUFRESNE
STELLUNG NACH 19. TA1 - D1 !
19. ...       D*F3      20. T*E7+!!   S*E7
21. D*D7+!! K*D7        22. LF5++      KE8
23. LD7+     KD8        24. L*E7 MATT!
GESPIELT 1852 IN BERLIN
```



Schachgrafiken mit dem 1520-Plotter



Der Text kann frei eingegeben werden

andere Seite gespiegelt. Außerdem wird der letzte Punkt mit dem ersten verbunden. Normalerweise werden die Punkte vom Programm direkt verbunden, also wird die Figur ohne Absetzen gezeichnet. In manchen Fällen erscheint dies aber nicht sinnvoll, zum Beispiel beim Auge des Springers oder den waagerechten Turmverstrebungen. Dabei bedient man sich der Technik des »NEUEN ANFANGSPUNKT SETZEN«, das Programm reagiert nach Joystick rechts + Feuer mit einem Ton und Punktkoordinaten 99,99. Nun kann der neue Anfangspunkt angefahren werden.

Bei über 40 Punkten werden die folgenden wieder rechts oben angezeigt.

Diagrammdrucker (Listing 1): Nach Starten dieses Programmes muß zuerst die Stellung auf die übliche Art (König: C1 und so weiter) eingegeben werden. Dies geschieht durch eine mit GET für diese Anwendung geschriebene Eingaberoutine. Eine Eingabe schließt mit mit <RETURN> ab. Gelöscht werden kann mit <DEL>. Durch <-> erkennt das Programm, daß von der einzugebenden Figur alle eingegeben worden sind.

Auch für Kassette

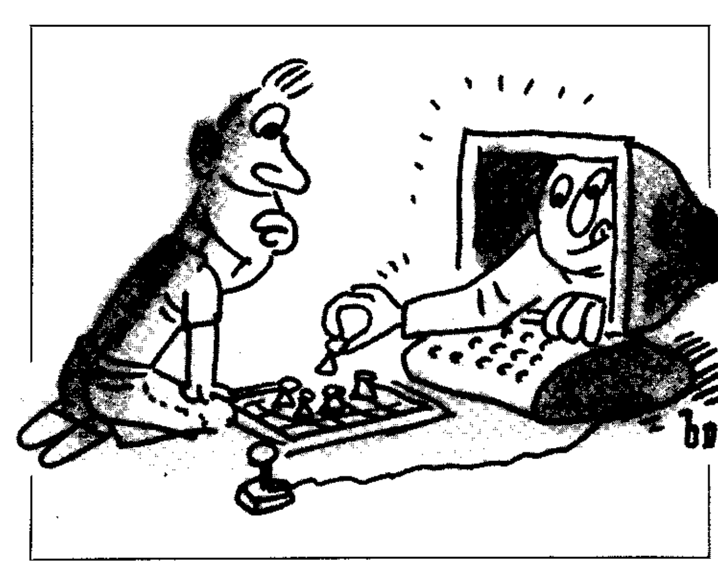
Nach Eingabe der Position kann man wählen, ob man sein Diagramm mit Text drucken will. Weiterhin kann man entscheiden, ob ein Standardmuster (Turnier, etc.), das für Turnierschach besonders geeignet ist, gewählt werden soll. Der Text kann nun frei auf dem Bildschirm (bis zum Balken) eingegeben werden und wird mit <RETURN> abgeschlossen. Nach dem Drucken erscheint eine Auswahl der erstellten Figurensätze, deren Namen ab Zeile 2000 eingegeben werden müssen und mit <E> abgeschlossen werden sollten. Nach dem Laden der Daten wird das Diagramm gedruckt.

Kassettenhilfe: Das Programm ist eigentlich für das Diskettenlaufwerk erdacht, doch mit folgender Methode läßt sich ein geeignetes File auf Datasette erzeugen:

1. Für jede Figur ein File »Figuren« erstellen.
2. Diese sechs Files zu einem zusammenfassen. Diese Aufgabe könnte Listing 3 »Tape-Hilfe« übernehmen.

Anpassung an den C 64: Der Satzersteller könnte mit einer Basic-Erweiterung die Befehle für hochauflösende Grafik (zum Beispiel Simons Basic) leicht umgeschrieben und dabei dank des größeren Speicherplatzes komfortabler gestaltet werden. Der Diagrammdrucker läßt sich leicht im C 64-Basic für den C 64 umschreiben. Lediglich einige GETKEY, IF THEN ELSE und ein RESTORE X müßten umgeformt werden. DO LOOP oder andere Basic-3.5-Konstruktionen sind in keinem Programm benutzt worden. Etwas unübersichtlichere FOR-NEXT- und IF-THEN-Konstruktionen sind zur besseren Anpassung an den C 64/VC 20 verwendet worden.

(M. Wesner/og)



Einsteins Geburtstag

Wollen Sie wissen, ob Einstein ein Sonntagskind war? Mit dem Programm »Wochentag« können Sie es feststellen.

Oft weiß man das Datum von einem besonderen Tag, aber nicht den dazugehörigen Wochentag. Beim Programm »Wochentag« brauchen Sie nur das entsprechende Datum eingeben und schon wissen Sie den dazugehörigen Wochentag. Selbstverständlich werden die Schaltjahre dabei berücksichtigt.

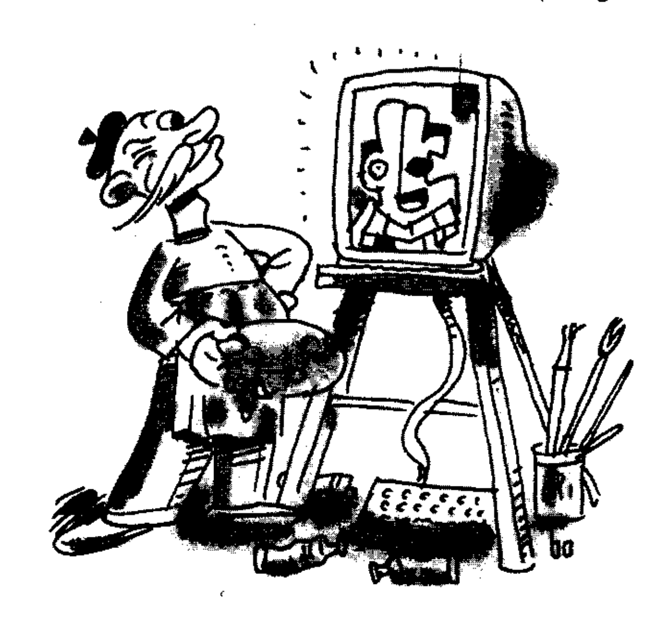
Nachdem Sie das Listing eingegeben und das Programm mit »RUN« gestartet haben, können Sie eine Notiz eingeben, wie beispielsweise einen Namen. Später erscheint diese Eingabe links oben am Bildschirm. Wollen Sie auf eine Notiz verzichten, so drücken Sie einfach <RETURN>.

Auf der nun folgenden Bildschirmmaske wird rechts oben die Zeit in Stunden, Minuten und Sekunden angegeben. Diese Angabe erfolgt einmalig beim Anspringen der Bildschirmmaske. Vor dem Starten des Programms können Sie mit TI$ = "....." (6stellige Zahl für Stunden, Minuten und Sekunden) die Zeit einstellen. Haben Sie dies nicht getan, so können Sie mit der Zeitangabe die jeweiligen Bearbeitungszeiten feststellen.

Mit dem Format »TT.MM.JJJJ« geben Sie Tag, Jahr und Monat des gewünschten Datums ein. Das Jahr muß vierstellig eingegeben werden. Bis 1582 dürfen Sie zurück in die Geschichte greifen. Nach <RETURN> wird das eingegebene Datum auf Richtigkeit überprüft. War alles richtig, so wird anschließend der zugehörige Wochentag auf dem Bildschirm angezeigt.

Finden Sie doch einmal heraus, wer aus Ihrer Verwandtschaft ein Sonntagskind ist oder ob Vater, Großvater und Urgroßvater vielleicht am gleichen Tag geboren wurden. Sie können auch jetzt schon herausfinden, an welchem Wochentag Sie Ihren 50. Geburtstag feiern werden.

(J.Hagen/kn)



```
10 REM  ERMITTLUNG DER WOCHENTAGE  VON  J. HAGEN
30 B$="SAMSTAG{3SPACE}SONNTAG{3SPACE}MONTAG{4SPA
   CE}DIENSTAG{2SPACE}MITTWOCH{2SPACE}DONNERSTAG
   FREITAG{3SPACE}"
40 P$="{8SPACE}": A$=P$: O$=CHR$(145): L$=CHR$(1
   57)
60 SCNCLR : PRINT : PRINT : INPUT "{5SPACE}NOTIZ
   : ";A$
90 DO
100 SCNCLR : D$="TT.MM.JJJJ": PRINT : PRINT " ";
    A$;
130 PRINT TAB(31) MID$(TI$,1,2);":";MID$(TI$,3,2
    );":";MID$(TI$,5,2): PRINT : PRINT
180 PRINT TAB(07)"ERMITTLUNG DES WOCHENTAGES"
190 PRINT TAB(06)"===========================":
     PRINT : PRINT : PRINT
210 PRINT : PRINT TAB(11)"DATUM{3SPACE}";D$;
230 FOR X=1 TO 12: PRINT L$;: NEXT X
260 INPUT D$
270 MID$(P$,1,2)=MID$(D$,1,2)
280 MID$(P$,3,2)=MID$(D$,4,2)
290 MID$(P$,5,4)=MID$(D$,7,4)
300 FOR K=1 TO 7
310 IF ASC(MID$(P$,K,1)) < 48 OR ASC(MID$(P$,K,1
    )) > 57 THEN 740
320 NEXT K
330 T=VAL(MID$(P$,1,2))
340 M=VAL(MID$(P$,3,2))
350 J=VAL(MID$(P$,5,4))
360 IF J<1582 THEN 740
370 IF M<1 OR M>12 THEN 740
380 IF T<1 OR T>31 THEN 740
390 IF M=4 AND T>30 THEN 740
400 IF M=6 AND T>30 THEN 740
410 IF M=9 AND T>30 THEN 740
420 IF M=11 AND T>30 THEN 740
430 IF M=2 AND T>29 THEN 740
440 IF M=2 AND T<29 THEN 480
450 IF M<>2 THEN 480
460 A=INT(J/4): B=A*4: IF J<>B THEN 740
480 GOSUB 790
500 PRINT O$;: PRINT TAB(11)"DER{5SPACE}";
510 PRINT MID$(P$,1,2);".";MID$(P$,3,2);".";MID$
    (P$,5,4)
540 PRINT : PRINT : PRINT TAB(08)"IST / WAR{2SPA
    CE}EIN{2SPACE}";MID$(B$,W*10+1,10): PRINT :
    PRINT
580 PRINT : PRINT : PRINT TAB(12)"WEITER J / N{3
    SPACE}J";L$;L$;L$;: INPUT C$
620 IF C$="J" GOTO 690
630 IF C$="N" GOTO 690
650 PRINT : PRINT TAB(08)"UNGUELTIGE EINGABE !";
    O$;O$;O$;O$;O$: GOTO 580
690 LOOP UNTIL C$ = "N"
710 PRINT : PRINT TAB(08) "{20SPACE}": PRINT
730 END
740 PRINT : PRINT : PRINT TAB(08)"UNGUELTIGES DA
    TUM{2SPACE}!";O$;O$;O$;O$;: GOTO 210
790 REM RECHNEN
800 U=365*J+31*(M-1)+T: IF M>2 THEN 830
820 GOTO 860
830 Y=INT(0.4*M+2.3): V=U-Y: GOTO 870
860 V=U: J=J-1
870 S=INT(J/4): T=INT(J/100): T=INT((T+1)*0.75):
     S=S-T: X=V+S
920 G=INT(X/7): W=X-G*7: RETURN
```

Listing. Das Programm »Wochentag« errechnet für ein Datum den zugehörigen Wochentag.
Beachten Sie bitte die Eingabehinweise auf Seite 130.

Hypotheken berechnen

Wie teuer kommt Ihr Haus wirklich? Mit diesem Programm können Sie die tatsächlichen Kosten berechnen.

Sie wollen sich ein Haus, eine Eigentumswohnung oder etwas ähnliches zulegen? Mit dem Programm »Hypothek« (Bild) können Sie die Kosten durch Hypotheken oder Kredite schnell ermitteln. Sicherlich werden Sie nicht jeden Einzelfall durch dieses relativ kurze Programm abdecken können. Doch zumindest erhalten Sie einen ersten Überblick über die Belastungen.

Das Programm ermittelt für einen bestimmten Kredit- oder Hypothekbetrag nebst dem dazugehörigen Zins- und Tilgungssatz die Monatsraten. Anschließend werden die Zins- und Tilgungsbeträge und der noch offene Kredit für jedes Jahr berechnet, bis die endgültige Tilgung erreicht ist.

Kommen wir nun zum Programm selbst. Wenn Sie das Listing eingegeben und das Programm mit <RUN> gestartet haben, können Sie zunächst eine Notiz (Name, Datum, etc.) eingeben. Ist dies nicht erforderlich, dann drücken Sie <RETURN>. Nun kommen Sie zu der eigentlichen Bildschirmmaske des Programms. Oben rechts finden Sie eine Zeitangabe. Sie können entweder vor dem Programmstart über TI$ die tatsächliche Zeit eingeben, oder mit der Angabe im Bildschirm die Bearbeitungszeit feststellen.

Als erstes werden Sie nun nach der Kredithöhe gefragt. Es folgen der Zins- und Tilgungssatz. Alle Eingaben müssen numerisch und (außer dem Zinssatz) größer Null sein. Mit dem Zinssatz ist der Effektivzins gemeint. Wenn Sie einen Tilgungssatz von 1,14% eingeben, ergibt sich übrigens eine Laufzeit von 30 Jahren für den Kredit.

Nach der vollständigen Eingabe erscheint auf dem Bildschirm die zu zahlende Monatsrate. Darunter werden die jährlichen Werte ausgegeben. Dabei überschreiben die neuen Jahreswerte die alten. Um die Übersicht zu behalten, geschieht dies mit einer zeitlichen Verzögerung. Sobald der offene Kredit den Wert Null erreicht hat, erscheinen die Summen und die Dauer der Rückzahlung.

Mit der Taste <RETURN> können Sie einen neuen Bearbeitungsvorgang aufrufen. Ein »N« für Nein beendet das Programm.

(J. Hagen/kn)

```
 H Y P O T H E K                    00:00:57
 ================
 KREDIT-HOEHE      ? 250000
 ZINS-SATZ  IN %  ? 6.66
 TILGUNG    IN %  ? 1.14
 MONATS-RATE   =    1625

      ZINSEN        TILGUNG         KREDIT
    1,011.97      15,194.73           0.00
 ------------  ------------
  331,706.70    250,000.00     30 JAHR(E)
 ============  ============

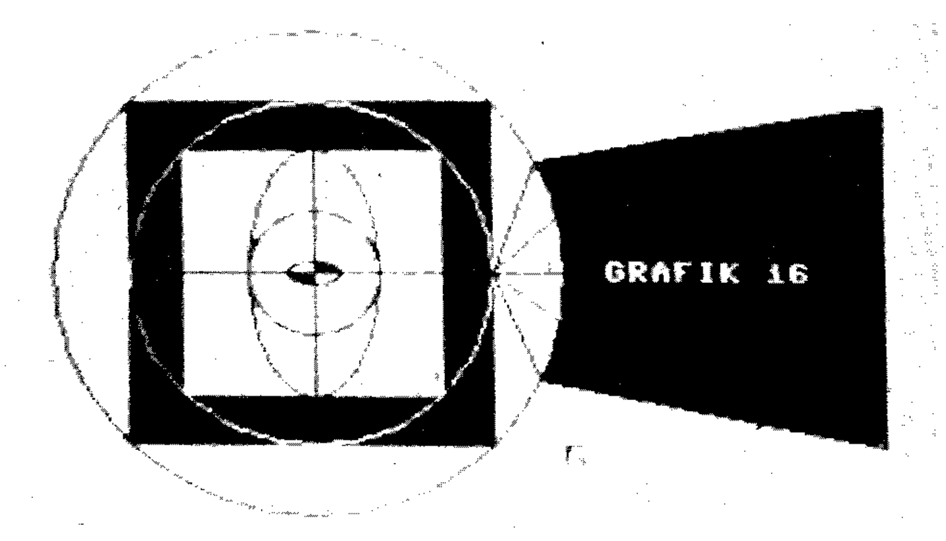
         NOCH EINMAL  J / N ? J
```

Bild. Das Programm »Hypothek«. Ein Kredit über 250000 Mark wurde hier berechnet.

```
10 REM          HYPOTHEK   VON J. HAGEN
20 P$="{BSPACE}": D$=P$: O$=CHR$(145): L$=CHR$(1
   57)
30 SCNCLR : PRINT : PRINT : INPUT "{5SPACE}NOTIZ
   : ";D$
40 DO
50 TG=0: ZG=0: A=0: T=0: Z=0: K=0
60 SCNCLR : PRINT : PRINT " ";D$;
70 PRINT TAB(31) MID$(TI$,1,2);":";MID$(TI$,3,2)
   ;":";MID$(TI$,5,2)
80 PRINT TAB(12)"H Y P O T H E K"
90 PRINT TAB(11)"=================": PRINT
100 PRINT TAB(11)"KREDIT-HOEHE{3SPACE}";: INPUT
    K
110 IF K<9999999.01 THEN 130
120 PRINT : PRINT TAB(06)"KREDIT ZU HOCH !";: GO
    TO 150
130 IF K>0 THEN 160
140 PRINT : PRINT TAB(06)"KREDIT NICHT > 0";
150 PRINT O$;O$;O$: GOTO 100
160 PRINT
170 PRINT TAB(06)"{5SPACE}ZINS-SATZ IN % ";: INP
    UT Z
180 IF Z<0 THEN 220
190 IF Z>100 THEN 210
200 GOTO 240
210 PRINT : PRINT TAB(06)"ZINS IST ZU HOCH";: GO
    TO 230
220 PRINT : PRINT TAB(06)"ZINS IST NEGATIV";
230 PRINT O$;O$;O$: GOTO 170
240 PRINT
250 PRINT TAB(06)"{5SPACE}TILGUNG{3SPACE}IN % ";
    : INPUT T
260 IF T>100 THEN 290
270 IF T>0 THEN 320
280 GOTO 300
290 PRINT : PRINT TAB(06)"TILGUNG{2SPACE}ZU HOCH
    ";: GOTO 310
300 PRINT : PRINT TAB(06)"TILGUNG NICHT > 0";
310 PRINT O$;O$;O$: GOTO 250
320 JR=K*(Z+T)/100: OK=K
330 PRINT : PRINT TAB(06)"{5SPACE}MONATS-RATE{2S
    PACE}={2SPACE}";JR/12
340 FOR X=1 TO 1000: NEXT X
350 PRINT : PRINT : PRINT "{7SPACE}ZINSEN{6SPACE
    }TILGUNG{7SPACE}KREDIT": PRINT
360 FOR X=1 TO 500: NEXT X
370 BZ=OK*Z/100: BT=JR-BZ: OK=OK-BT: TG=TG+BT: Z
    G=ZG+BZ
380 PRINT USING "##,###,###.##";BZ;
390 PRINT USING "##,###,###.##";BT;
400 PRINT USING "##,###,###.##";OK: A=A+1
410 FOR X=1 TO 300: NEXT X
420 IF OK<=0 THEN 460
430 PRINT O$;O$: IF OK<JR THEN 450
440 GOTO 370
450 JR=OK: BZ=OK*Z/100: BT=OK: TG=TG+BT: ZG=ZG+B
    Z: OK=OK-BT: GOTO 380
460 PRINT " ------------ ------------"
470 PRINT USING "##,###,###.##";ZG;
480 PRINT USING "##,###,###.##";TG;
490 PRINT "{2SPACE}";A;"JAHR(E)"
500 PRINT " ============ ============"
510 PRINT : PRINT TAB(09)"{20SPACE}"
520 PRINT TAB(09)"NOCH EINMAL{2SPACE}J / N{3SPAC
    E}J";L$;L$;L$;: INPUT F$
530 IF F$="J" GOTO 560
540 IF F$="N" GOTO 560
550 PRINT : PRINT TAB(09)"UNGUELTIGE EINGABE !";
    O$;O$;O$: GOTO 520
560 LOOP UNTIL F$="N"
570 PRINT : PRINT TAB(09)"{20SPACE}";O$;O$;O$
580 END
```

Listing. »Hypotheken berechnen«.
Bitte beachten Sie die Eingabehinweise auf Seite 130.

Grafik leichtgemacht



Ein Bild, das mit »Grafik 16« erstellt wurde

Viel Platz zum Programmieren bleibt nicht mehr, wenn Sie beim C16 im Grafik-Modus arbeiten. Dennoch bietet »Grafik 16« einigen Komfort, wenn Sie Grafiken erstellen wollen.

Zwei KByte Basic-Speicher bei eingeschalteter Grafik sind nicht viel. Natürlich kann man kein Super-Programm wie etwa Hi-Eddi für den C64 in einem so kleinen Speicherbereich unterbringen. Aber mit den komfortablen Grafikbefehlen des C16 läßt sich doch schon allerhand anfangen, wie Sie in der Abbildung sehen können.

Programmübersicht	
Zeile 10	Dimensionierung. Datasette oder Floppy? HiRes oder Multicolor?
Zeile 20	Anfangswerte einstellen.
Zeile 30 bis 50	Pfeil und Radiergummi zeichnen und als Shapes speichern.
Zeile 50	Aktuelles Shape auf den Bildschirm bringen.
Zeile 60	Tastaturabfrage und Shape löschen.
Zeile 70 bis 190	Befehle auf Grenzwerte testen und ausführen. Shape wieder auf den Bildschirm bringen.
Zeile 200	Nach <P> Shape auf Speicherposition bringen und nach Tastendruck wieder löschen.
Zeile 210 bis 250	Speichern und Laden von Bildern über den Monitor, gegebenenfalls Name eingeben.
Zeile 260 bis 360	Ändern der Farben nach <W>. Zeile 270 Aktuelle Zeichenfarbe anzeigen. Zeile 280 Farb- und Luminanzwerte der Farbnummern einlesen. Zeile 290 Tastaturabfrage und Farbe anzeigen. Zeile 300 bis 360 Befehle auf Grenzwerte testen und ausführen. Farbnummern neue Farben zuordnen.
Zeile 370	Taste <STOP> abfangen.

Tabelle. Übersicht zu dem Programm »Grafik 16«

Bei dem Programm »Grafik 16« ist der verbleibende Basic-Speicher voll ausgenutzt. Versuchen Sie daher nicht, das Programm zu erweitern. Wenn Sie keine Speichererweiterung eingebaut haben, würde dies unweigerlich zum Absturz des Programms führen.

Kommen wir aber nun zur Beschreibung des Programms. Sie werden merken, daß einiges geboten wird.

Wenn Sie das Listing eingegeben und das Programm mit »RUN« gestartet haben, tut sich zunächst einmal gar nichts. Sie müssen nun zuerst wählen, ob Sie im HiRes- (Taste <1>) oder Multicolor-Modus (Taste <2>) arbeiten wollen. Im HiRes-Modus gibt es zwei Farben, während Ihnen bei Multicolor vier Farben zur Verfügung stehen. Beim Erstellen der Grafiken können Sie die Farben mit den Tasten <1> und <2> beziehungsweise <1> bis <4> anwählen. Vom Programm wird anfangs die Farbe 1 eingestellt.

Grafik in Farbe

Über die Taste <W> können Sie einer Farbnummer auch verschiedene Farben zuordnen. Nach dem Betätigen der Taste erscheint unten ein Fenster mit der aktuellen Zeichenfarbe. Darunter ist die Nummer einer Farbe neben der entsprechenden Farbprobe eingeblendet. Mit den Cursor-Tasten <Links> und <Rechts> können Sie die Farbnummer aufrufen, durch <Hoch> und <Runter> ordnen Sie eine beliebige Farbe zu. Mit <+> und <-> wird die Helligkeit (Luminanz) aller Farben verändert. Wenn Sie eine andere Farbe auch für das Bild übernehmen wollen, dann drücken Sie <RETURN>. Durch <STOP> kehren Sie zurück zu Ihrem Grafikbildschirm.

Mit den Cursor-Tasten können Sie den Malpfeil nach oben, unten, rechts und links bewegen. Aber auch diagonale Positionsveränderungen sind möglich. Mit <@> nach oben-links, <+> nach oben-rechts, <*> nach unten-rechts und mit <;> nach unten-links.

Nach dem Starten des Programms verändert der Malpfeil seine Position je Tastendruck nur pixelweise. Eine sehr genaue Positionierung ist dadurch möglich. Sie können aber auch Sprünge von bis zu 20 Pixeln hervorrufen. Mit der Taste <.> werden die Sprünge größer, mit <,> kleiner. Über

<SHIFT+.> und <SHIFT+,> können Sie direkt zwischen 20 und 1 Pixel wechseln. In der Praxis erweist sich dies als sehr nützlich.

Mit <HOME> können Sie die aktuelle Position speichern. Eine Linie von der letzten Speicherposition zur derzeitigen Position des Malpfeils wird mit <L> gezogen. Recht hilfreich ist: über <P> können Sie stets die letzte Speicherposition abfragen. Nach einem beliebigen Tastendruck ist der Malpfeil wieder an der vorherigen Stelle.

Nach <V> wird ein Viereck zwischen der Speicherposition und dem Malpfeil gezeichnet. Ein <Q> anstelle von <V> sorgt dafür, daß gleich ein ausgefülltes Viereck gezeichnet wird. Ein Kreis um die Speicherposition entsteht, wenn Sie <K> drücken. Der Radius entspricht dem Abstand zwischen Speicherposition und Malpfeil. <O> erzeugt eine Ellipse. Die beiden Radien entsprechen dabei den Kantenlängen jenes Vierecks, das Sie in dieser Position aufrufen könnten.

Viele Funktionen

Mit <Z> gelangen Sie in den sogenannten Zeichen-Modus. Hier wird an jeder neuen Position des Malpfeils ein Punkt gesetzt. Wenn Sie den Pfeil nur pixelweise wandern lassen, entsteht so eine Linie. Durch <ESC> verlassen Sie diesen Modus.

Durch <T> ist es möglich, Text in die Zeichnung einzubringen. Der Text muß in ein zunächst erscheinendes Fenster eingegeben werden. Der Malpfeil befindet sich übrigens am linken Textrand, in der Mitte der Texthöhe. Geben Sie den Text aus dem Zeichen-Modus ein, so erscheint er anschließend revers.

Über den Befehl <F> können Sie geschlossene Bereiche ausfüllen. Dieser Vorgang läßt sich mit der Taste <STOP> abbrechen. Achten Sie darauf, daß dieser Befehl sich nur auf Flächen bezieht, die in der gleichen Farbe eingegrenzt wurden. Wollen Sie beispielsweise einen in der Farbe 1 eingegebenen Kreis mit der Farbe 2 füllen, so kann eventuell der ganze Bildschirm mit der Farbe 2 ausgefüllt werden. Auch wenn Sie einer Farbnummer während Ihrer Arbeit über <W> eine andere Farbe zuordnen, können Schwierigkeiten auftreten.

Wenn Sie die Taste <R> drücken, erscheint auf dem Bildschirm ein Radiergummi. Nachdem dieser Befehl aufgerufen wurde, ist die Schrittweite zunächst automatisch auf 1 Pixel eingestellt, um kleine Bereiche löschen zu können. Mit <ESC> verlassen Sie den Lösch-Modus.

Bilder speichern

Vorsichtig sollten Sie mit dem Befehl <SHIFT + CLEAR/HOME> umgehen, denn damit wird der gesamte Bildschirm gelöscht. Verwechseln Sie ihn bitte nicht mit »Position speichern«, wobei die gleiche Taste ohne <SHIFT> benutzt wird.

Als letztes kommen wir dazu, wie Bilder gespeichert und geladen werden. Dies erfolgt mit <C= + S> für Speichern und <C= + L> für Laden. Nach dem Aufruf des Befehls können Sie jeweils den Dateinamen eingeben. Das Laden und Speichern erfolgt über den eingebauten Monitor »TEDMON«. In der Zeile 10 des Programms können Sie übrigens mit der Variablen »K« die Gerätenummer einstellen (1 = Kassette, 8 = Diskette).

Das ganze Programm wird mit <CTRL> <C> beendet.

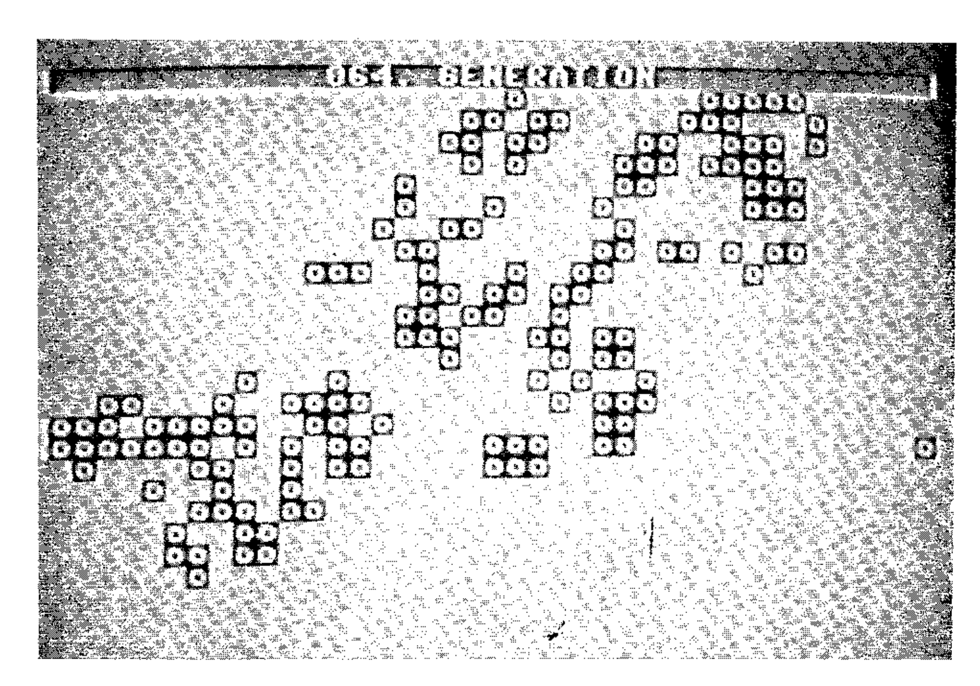
Eine Übersicht, was in welchen Zeilen des Programms gemacht wird, erhalten Sie in der nebenstehenden Tabelle.

(Jörg Gerjets/kn)

```
10 K=8: DIM F%(4),L%(4),F$(1): GET KEY P: IF P=1
   THEN G=1: ELSE G=3
20 GRAPHIC G,1: X=320/P/2: Y=100: S=1: W=4: U=1:
   D$="{BLACK,CLR,20DOWN}"
30 DRAW G,5/P,0 TO 0,0 TO 0,5: DRAW G,0,0 TO 7/P
   ,7: SSHAPE F$(0),0,0,7/P,7
40 BOX G,9,1,14,6,,1: BOX 0,11,2,12,5,,1: SSHAPE
   F$(1),8,0,15,7: SCNCLR
50 GSHAPE F$(F),X+1,Y+1,4
60 DO : TRAP 370: GRAPHIC G: GET KEY A$: GSHAPE
   F$(F),X+1-F,Y+1-F,4: IF W=0 THEN S=1
70 TRAP 50: IF(A$="{UP}" OR A$="@" OR A$="+") AN
   D Y>S-1 THEN Y=Y-S: ELSE IF A$="R" THEN F=1:
   W=0
80 IF(A$="{LEFT}" OR A$="@" OR A$=";") AND X>S/P
   -1 THEN X=X-S/P: ELSE IF A$="," AND S>1 THEN
   S=S-1
90 IF(A$="{RIGHT}" OR A$="+" OR A$="*") AND X<32
   0/P-S/P THEN X=X+S/P
100 IF A$>"/" AND ASC(A$)<49+G THEN U=VAL(A$)
110 IF(A$="{DOWN}" OR A$=";" OR A$="*") AND Y<20
    0-S THEN Y=Y+S: ELSE IF A$="V" THEN BOX U,A,
    B,X,Y
120 IF A$="F" THEN PAINT U,X,Y: ELSE IF A$="{CLR
    }" THEN SCNCLR : ELSE IF A$=" " OR D=1 THEN
    DRAW U,X,Y
130 IF A$="K" THEN CIRCLE U,A,B,SQR(ABS(X-A)↑2+A
    BS((Y-B)/P)↑2): ELSE IF A$="<" THEN S=1
140 IF A$="{HOME}" THEN A=X: B=Y: ELSE IF A$="S"
     OR A$="L" THEN 210: ELSE IF A$="{CTRL+C}" T
    HEN GRAPHIC 0: PRINT D$: END
150 IF A$=">" THEN S=20: ELSE IF A$="." AND S<20
     THEN S=S+1: ELSE IF A$="L" THEN DRAW U,A,B
    TO X,Y
160 IF A$="O" THEN CIRCLE U,A,B,ABS(X-A),ABS(Y-B
    ): ELSE IF A$=CHR$(27) THEN D=0: F=0: W=4
170 IF A$="W" THEN TRAP 50: GOSUB 260: ELSE IF A
    $="Q" THEN BOX U,A,B,X,Y,,1: ELSE IF A$="Z"
    THEN D=1
180 IF A$="T" THEN GRAPHIC G+1: A$="": PRINT D$:
     INPUT A$: CHAR U,X/(8/P),Y/8,A$,D: D=0
190 IF A$<>"P" THEN GSHAPE F$(F),X+1-F,Y+1-F,W:
    LOOP
200 GSHAPE F$(0),A+1,B+1,4: GET KEY A$: GSHAPE F
    $(0),A+1,B+1,4: GOTO 50
210 GRAPHIC 0,1: IF K=8 OR A$="S" THEN INPUT " N
    AME";N$
220 PRINT "{CLR}MQ{5DOWN}": IF A$="L" THEN PRINT
     "L";: IF K=8 THEN PRINT CHR$(34)N$ CHR$(34)
    ","K;
230 IF A$="S" THEN PRINT "S" CHR$(34)N$ CHR$(34)
    ","K"{LEFT},1800,4000";
240 PRINT : PRINT "{6DOWN}X": PRINT "{DOWN}GQ50"
250 POKE 1319,19: FOR I=1320 TO 1328: POKE I,13:
     NEXT : POKE 239,10: END
260 IF RCLR(0)=1 THEN W$="{WHITE}": ELSE W$="{BL
    ACK}"
270 PRINT D$W$"{RVOFF}ZEICHENFARBE:";: PRINT U:
    PRINT : GRAPHIC G+1
280 FOR I=0 TO 4: F%(I)=RCLR(I): L%(I)=(I): NEXT
     : I=1: GOSUB 360
290 DO : PRINT "{RVOFF}"W$;I;: COLOR 1,C,L: PRIN
    T "{RVSON,11SPACE,14LEFT}";: GET KEY A$
300 IF A$="{LEFT}" AND I>0 THEN I=I-1: GOSUB 360
     : ELSE IF A$="{RIGHT}" AND I<4 THEN I=I+1: G
    OSUB 360
310 IF A$="{DOWN}" AND C>1 THEN C=C-1: ELSE IF A
    $="{UP}" AND C<15 THEN C=C+1
320 IF A$="+" AND L>0 THEN L=L-1: ELSE IF A$="-"
     AND L<7 THEN L=L+1
330 IF A$="*" THEN 350: ELSE IF A$<>CHR$(13) THE
    N LOOP
340 F%(I)=C: L%(I)=L: LOOP
350 FOR I=0 TO 4: COLOR I,F%(I),L%(I): NEXT : RE
    TURN
360 C=F%(I): L=L%(I): RETURN
370 RESUME
```

Listing zu »Grafik 16«.
Bitte beachten Sie die Eingabehinweise auf Seite 130.

Das Spiel des Lebens



»Life« ist eine »Lebenssimulation«, deren Regeln für fast jeden Computertyp umgesetzt wurden. Diese Version ist auf dem C16, C116 und dem Plus/4 lauffähig. Versuchen Sie sich als Evolutionsforscher, »Life« macht's möglich.

Spielregeln: Die Spielfläche (Lebensraum) ist eine in Felder aufgeteilte Ebene (24 x 40 Felder). Jedes Feld kann zwei Zustände, nämlich unbesetzt (tot) und besetzt (lebend) annehmen. Es gelten folgende Übergangsregeln:

- Ein leeres Feld geht in ein besetztes Feld über, wenn, genau drei seiner Nachbarfelder besetzt sind.
- Ein besetztes Feld geht in ein leeres Feld über, wenn weniger als zwei oder mehr als drei seiner Nachbarfelder besetzt sind, andernfalls bleibt es besetzt.

Das Programm verfügt über zwei Geschwindigkeitsstufen. Nach Eingabe einer Anfangsgeneration gelangt man durch Drücken der <F1>-Taste in den Fast-Modus. Hier werden zirka 5 Generationen pro Sekunde berechnet und auf dem Bildschirm angezeigt. Mit der <F2>-Taste gelangt man in den Step-Modus. Erst nach Drücken der <SPACE>-Taste wird die nächste Generation gezeigt. Mit der <F3>-Taste kann das Programm gestoppt werden. Man befindet sich danach wieder in der Eingaberoutine. Die letzte Generation bleibt dabei erhalten, sie kann nun abgeändert werden, um danach das Programm wieder mit <F1> oder <F2> zu starten. Die <HELP>-Taste bewirkt dasselbe wie die <F3>-Taste, nur wird hierbei der Lebensraum gelöscht.

In der Eingaberoutine können Elemente mit der <*>-Taste gesetzt und mit der <INST/DEL>-Taste gelöscht werden. Mit der <RETURN>-Taste wird das Programm beendet.

Das Programm kann zwar mit dem eingebauten Monitor eingegeben werden, aber die Wahrscheinlichkeit, daß man 1430 Zahlen richtig eingibt, ist wohl sehr gering. Deshalb ist nur zu empfehlen das Programm mit dem Eingabeprogramm (Listing 1) einzugeben.

Mit Hilfe dieses Programms kann das Hexadezimal-Listing (Listing 2) schneller und vor allem richtig eingegeben werden. Nach Eingabe von jeweils 8 Byte (eine Zeile im Listing) erscheint eine Prüfnummer. Diese Prüfnummer kann mit der Prüfnummer auf dem Hex-Listing verglichen werden. Sind die Prüfnummern gleich, drückt man die <J>-Taste; es können dann die nächsten 8 Byte eingegeben werden. Sind die Prüfnummern ungleich, hat man bei der Eingabe einen Fehler gemacht. In diesem Fall drückt man die <N>-Taste und gibt die letzten 8 Byte noch einmal ein. Das Eingabeprogramm darf natürlich nicht den gleichen Speicherbereich benutzen wie das Life-Programm. Deshalb setzt man durch Eingabe von

```
POKE 43,1:POKE 44,32:POKE 8192,0:NEW <RETURN>
```

den Basic-Speicherbeginn hinter dem Life-Programm. Erst jetzt darf das Eingabeprogramm eingetippt werden.

Damit das Life-Programm genauso geladen und gestartet werden kann wie ein Basic-Programm, sollte man nach Eingabe des Programms wie folgt vorgehen:

Reset-Taste drücken (alle Zeiger werden zurückgesetzt)

```
100 PRINT CHR$(147)
110 INPUT"ANF.-ADDR. ";A$
120 INPUT"END.-ADDR. ";E$
130 A=DEC(A$):E=DEC(E$)
140 FOR X=A TO E STEP 8
150 S=0:PRINT CHR$(147)
160 FOR Y=0 TO 7
170 PRINT HEX$(X+Y);"  ";
180 GETKEY A1$
190 PRINT A1$;
200 GETKEY A2$
210 PRINT A2$
220 A$=A1$+A2$:P=DEC(A$)
230 POKE X+Y,P
240 S=S+P*(Y+1)
250 NEXT Y
260 PRINT CHR$(17)
270 PRINT "PRUEFNR.: ";HEX$(S)
280 PRINT CHR$(17)
290 PRINT "RICHTIG (J/N)"
300 GETKEY A$
310 IF A$="N" THEN 150
320 NEXT X
```

Listing 1. Das Eingabeprogramm für »Life«. Bitte vor dem Eingeben verschieben (siehe Text).

1 SYS 4112 <RETURN> (Einsprungadresse des Life-Programms)
POKE45,168:POKE46,21 <RETURN> (die Zeiger für das Ende eines Programms werden auf das Ende des Life-Programms gesetzt)
SAVE "LIFE" <RETURN> (Programm wird gespeichert)

Danach kann das Programm mit LOAD geladen und mit dem Befehl RUN gestartet werden.

(Joachim Stolte/og)

```
1010   A9 93 20 D2 FF A9 7F 8D   =1649
1018   15 FF 8D 19 FF A9 42 A2   =13ED
1020   00 9D 00 08 9D 00 09 9D   =0992
1028   00 0A E8 D0 F4 EA EA EA   =1E02
1030   A0 00 A9 0B 85 3C A9 FE   =1357
1038   85 3B A2 00 BD 70 13 C9   =0FFF
1040   00 F0 11 18 65 3B 85 3B   =0B49
1048   90 02 E6 3C A9 A0 91 3B   =1112
1050   E8 18 90 E8 EA EA A2 11   =156C
1058   A0 0F 18 20 F0 FF A2 00   =109E
1060   BD F5 13 C9 00 F0 07 20   =0CD5
1068   D2 FF E8 18 90 F2 EA EA   =1C1A
1070   A9 10 85 3B A2 00 A0 00   =0ACE
1078   88 D0 FD CA D0 F8 C6 3B   =1969
1080   D0 F4 EA EA A2 00 BD 30   =12F3
1088   14 9D 5D 05 E8 E0 0F D0   =132A
1090   F5 EA EA EA A2 00 BD 40   =1384
1098   14 20 D2 FF E8 D0 F7 BD   =1CD7
10A0   40 15 C9 00 F0 07 20 D2   =0F0F
10A8   FF E8 18 90 F2 20 E4 FF   =1905
10B0   C9 85 D0 F9 A9 93 20 D2   =1656
10B8   FF A2 00 A9 20 9D 00 20   =0A35
10C0   9D 00 21 9D 00 22 9D 00   =088B
10C8   23 9D 00 24 9D 00 25 9D   =0AE9
10D0   00 26 9D 00 27 9D 00 28   =07D4
10D8   CA D0 E2 A2 28 A9 A0 9D   =159E
10E0   FF 0B A9 06 9D FF 07 CA   =12B4
10E8   D0 F3 A9 42 A2 F0 9D 27   =1406
10F0   08 9D 17 09 9D 07 0A 9D   =0A14
10F8   F7 0A CA D0 F1 A2 1F BD   =15EB
1100   81 15 9D FF 0B CA D0 F7   =18D9
1108   A2 07 BD 78 15 95 D8 CA   =14E6
1110   D0 F8 EA EA A9 F3 85 3C   =17A8
1118   A9 14 85 3B A9 0D 85 3D   =0C72
1120   A9 00 85 A9 A9 02 85 A8   =1118
1128   A9 02 85 AB A9 01 85 AA   =112E
1130   A0 00 18 A5 A8 69 B0 8D   =1272
1138   0F 0C A5 A9 69 B0 8D 10   =0F42
1140   0C A5 AA 69 B0 8D 17 0C   =0CB7
1148   A5 AB 69 B0 8D 18 0C B1   =0F23
1150   3C 09 80 91 3C A2 FF CA   =1653
1158   D0 FD 29 7F 91 3C A2 FF   =15E4
1160   CA D0 FD 18 A5 3C 85 3E   =0FF5
1168   A5 3D 69 14 85 3F EA 20   =0E23
1170   E4 FF C9 0D D0 02 60 EA   =137D
1178   C9 91 F0 1F C9 11 F0 1E   =110A
1180   C9 1D F0 1D C9 9D F0 1C   =1352
1188   C9 2A F0 1B C9 14 F0 1A   =101E
1190   C9 85 F0 19 C9 86 F0 18   =1368
1198   18 90 95 18 90 15 18 90   =0BCD
11A0   36 18 90 5C 18 90 78 4C   =0D06
11A8   3E 12 4C 4B 12 4C 56 12   =077E
11B0   4C 5D 12 A5 3D C9 0D 10   =0A92
11B8   06 A5 3C C9 50 90 15 C6   =10DB
11C0   AB 10 06 A9 09 85 AB C6   =11A9
11C8   AA 38 A5 3C E9 28 85 3C   =0EF9
11D0   B0 02 C6 3D 4C 30 11 A5   =0C35
11D8   3D C9 0F 90 07 A5 3C 18   =0AA1
11E0   C9 BF B0 19 E6 AB A5 AB   =1716
11E8   C9 0A D0 06 A9 00 85 AB   =0FAD
11F0   E6 AA 18 A5 3C 69 28 85   =0DF8
11F8   3C 90 02 E6 3D 4C 30 11   =09CB
1200   A5 3B C9 28 F0 16 E6 A9   =14DC
1208   A5 A9 C9 0A D0 06 A9 00   =0D4D
1210   85 A9 E6 A8 E6 3B E6 3C   =1533
1218   D0 02 E6 3D 4C 30 11 A5   =0CB5
1220   3B C9 02 30 12 C6 A9 10   =0CB0
1228   06 A9 09 85 A9 C6 A8 C6   =1640
1230   3B C6 3C A5 3C C9 FF D0   =186A
1238   02 C6 3D 4C 30 11 A0 00   =092B
1240   A9 D7 91 3E A9 51 91 3C   =100C
1248   4C 30 11 A0 00 A9 20 91   =0CBD
1250   3E 91 3C 4C 30 11 A9 A0   =0E39
1258   85 3B 18 90 05 A9 20 85   =0C9A
1260   3B EA EA A9 00 A2 03 95   =0FFA
1268   A7 CA D0 FB A2 1F BD 58   =1466
1270   15 9D 00 0C CA D0 F7 E6   =1842
1278   AA 18 A2 03 B5 A7 69 B0   =129E
1280   9D 0B 0C CA D0 F6 EA A0   =1939
1288   00 E6 AA A5 AA C9 0A F0   =162C
1290   02 D0 19 84 AA E6 A9 A5   =167A
1298   A9 C9 0A F0 02 D0 0D 84   =0F7E
12A0   A9 E6 A8 A5 A8 C9 0A D0   =15C5
12A8   02 84 A8 EA A2 F0 BD 27   =15D7
12B0   20 9D 27 0C BD 17 21 9D   =0C09
12B8   17 0D BD 07 22 9D 07 0E   =077D
12C0   BD F7 22 9D F7 0E CA D0   =16B2
12C8   E5 EA EA A5 C6 C9 03 F0   =1834
12D0   11 C9 06 F0 10 A5 3B 30   =0CC0
12D8   11 20 E4 FF C9 20 F0 0A   =1286
12E0   D0 E8 4C 94 10 4C FD 10   =0F57
12E8   EA EA A2 00 BD 00 20 9D   =0E1D
12F0   00 24 BD 00 21 9D 00 25   =07FA
12F8   BD 00 22 9D 00 26 BD 00   =09A6
1300   23 9D 00 27 CA D0 E5 EA   =185E
1308   EA A9 FF 85 3C 85 3E A9   =1291
1310   1F 85 3D A9 23 85 3F A9   =0F52
1318   28 85 27 A9 18 85 28 A9   =0E41
1320   00 85 26 A2 07 B4 D8 B1   =13CF
1328   3E 10 02 E6 26 CA 10 F5   =118E
1330   A0 29 A6 26 A9 20 91 3C   =0D60
1338   B1 3E 10 04 E0 02 F0 04   =0C89
1340   E0 03 D0 04 A9 D7 91 3C   =1194
1348   E6 3C D0 02 E6 3D E6 3E   =11FC
1350   D0 02 E6 3F C6 28 D0 C7   =1538
1358   C6 27 D0 BF 4C 79 12 EA   =12A0
1360   EA EA EA EA EA EA EA EA   =20E8
1368   EA EA EA EA EA EA EA EA   =20E8
1370   80 01 07 01 05 01 01 01   =00C9
1378   01 05 01 01 01 01 0D 01   =0080
1380   07 01 05 01 01 01 01 05   =0056
1388   01 01 01 01 0D 01 07 01   =008A
1390   05 01 08 01 10 01 07 01   =00B2
1398   05 01 08 01 10 01 07 01   =00B2
13A0   05 01 08 01 10 01 07 01   =00B2
13A8   05 01 01 01 01 05 01 01   =0040
13B0   01 01 0D 01 07 01 05 01   =0082
13B8   01 01 01 05 01 01 01 01   =0034
13C0   0D 01 07 01 05 01 08 01   =0087
13C8   10 01 07 01 05 01 08 01   =008A
13D0   10 01 07 01 05 01 08 01   =008A
```

Listing 2. »Life« für den C16.
Zur Eingabe verwenden Sie bitte Listing 1 (siehe Text).

```
13D8   10 01 01 01 01 04 01 05   =0065
13E0   01 08 01 01 01 01 0D 01   =0086
13E8   01 01 01 04 01 05 01 08   =0080
13F0   01 01 01 01 00 28 43 29   =0417
13F8   20 31 39 38 36 0D 0D 20   =04C4
1400   20 20 20 20 20 20 20 20   =0480
1408   20 20 20 20 20 20 20 20   =0480
1410   42 59 0D 0D 20 20 20 20   =048F
1418   20 20 20 20 20 20 20 20   =0480
1420   4A 4F 41 43 48 49 4D 20   =08F0
1428   53 54 4F 4C 54 45 00 00   =065A
1430   00 FF 01 01 01 00 00 00   =020A
1438   00 01 85 86 87 88 00 00   =097C
1440   93 90 20 20 20 20 20 20   =05D3
1448   20 20 20 20 20 20 20 47   =05B8
1450   41 4D 45 20 4F 46 20 4C   =0899
1458   49 46 45 0D 0D 20 20 20   =04B9
1460   20 20 20 20 20 20 20 20   =0480
1468   4E 41 43 48 20 4A 2E 48   =0897
1470   2E 20 43 4F 4E 57 41 59   =0A92
1478   0D 0D 0D 20 20 20 20 20   =040E
1480   20 20 20 20 20 20 20 41   =0588
1488   4E 4C 45 49 54 55 4E 47   =0AD5
1490   0D 0D 20 20 20 12 1C 46   =0507
1498   31 92 90 20 20 20 20 20   =06C5
14A0   46 41 53 54 2D 4D 4F 44   =0A09
14A8   55 53 0D 0D 20 20 20 12   =0426
14B0   1C 46 32 92 90 20 20 20   =08F6
14B8   20 20 53 54 45 50 2D 4D   =0985
14C0   4F 44 55 53 0D 0D 20 20   =0591
```

```
14C8   20 12 1C 46 33 92 90 20   =0B0B
14D0   20 20 20 20 53 54 4F 50   =0980
14D8   0D 0D 20 20 20 12 1C 48   =0517
14E0   45 4C 50 92 90 20 20 20   =0985
14E8   4E 45 55 45 20 45 49 4E   =0998
14F0   47 41 42 45 0D 0D 20 20   =0512
14F8   20 12 1C 53 50 41 43 45   =08F7
1500   92 90 20 20 53 54 45 50   =0A8C
1508   2D 54 41 53 54 45 0D 0D   =06E9
1510   20 20 20 12 1C 20 2A 20   =047A
1518   92 90 20 20 20 20 53 45   =085F
1520   54 5A 45 4E 0D 0D 20 20   =057E
1528   20 12 1C 44 45 4C 92 90   =0D47
1530   20 20 20 20 4C 4F 45 53   =0911
1538   43 48 45 4E 0D 0D 0D 20   =04C4
1540   20 20 20 20 20 20 20 53   =0618
1548   54 41 52 54 20 4D 49 54   =0A29
1550   20 12 1C 46 31 92 90 00   =0A01
1558   A0 A0 A0 A0 A0 A0 A0 A0   =1680
1560   A0 A0 A0 A0 A0 A0 A0 AE   =16F0
1568   A0 87 85 8E 85 92 81 94   =13A1
1570   89 8F 8E A0 A0 A0 A0 A0   =1611
1578   00 01 02 28 2A 50 51 52   =0821
1580   00 A0 A0 A0 A0 A0 A0 A0   =15E0
1588   A0 A0 A0 A0 A0 A0 A0 98   =1640
1590   BD A0 A0 A0 A0 A0 A0 99   =1665
1598   BD A0 A0 A0 A0 A0 A0 A0   =169D
15A0   A0 A0 A0 A0 A0 A0 A0 A0   =1680
```

Listing 2. »LIFE« für den C 16. (Schluß)

Gefräßige Riesenschlange

Stillen Sie den Hunger einer nimmersatten Schlange. Aber es ist Vorsicht geboten, denn sie frißt auch sich selbst!

Sie schlüpfen bei diesem bekannten Spiel in die Rolle einer Schlange, die einen schier unstillbaren Hunger auf grüne Äpfel hat (siehe Bild). Nach jedem verspeisten Kernobst wächst die Schlange etwas. Dadurch wird es im Spielverlauf immer schwieriger, an sich selbst vorbeizukommen. Auch die Mauer, die das Spielfeld eingrenzt, ist ungenießbar. Für jeden gescheiterten Versuch wird Ihnen ein Leben abgezogen. Sie haben insgesamt fünf Versuche, um in die nächsthöhere Spielstufe zu gelangen. Hierzu müssen Sie aber eine bestimmte Schlangenlänge erreichen, die mit dem Schwierigkeitsgrad wächst. Als Belohnung erhalten Sie einen Punktebonus, der von der erreichten Länge abhängt, und ein weiteres Leben. Allerdings wird in jeder neuen Spielphase auch der Schwierigkeitsgrad erhöht. »Snake« ist also ein spannendes Geschicklichkeitsspiel für reaktionsschnelle Spieler.

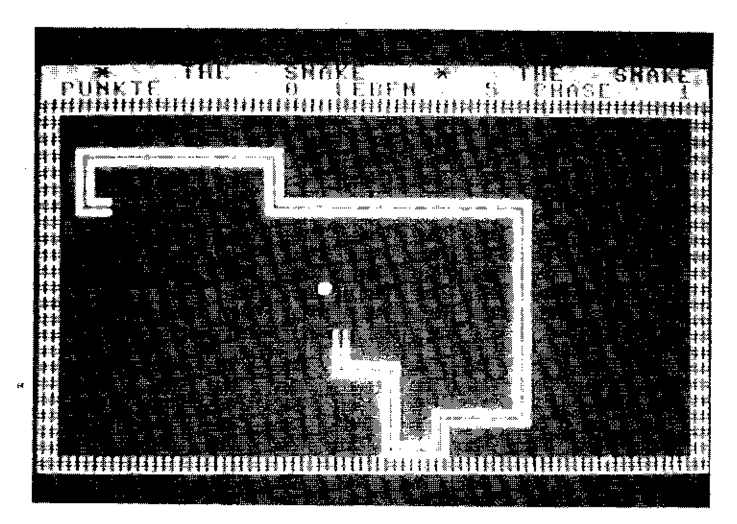


Bild. Spielsituation bei »Snake«. Die Schlange muß trotz wachsender Länge alle Äpfel fressen und darf sich weder über den eigenen Schwanz fahren noch gegen die Mauer stoßen.

Eingabe des Programms:

Bei der Programmierung des Spiels wurde ausgenutzt, daß sich der Joystick auch mit GET abfragen läßt. Deshalb kann anstelle eines Joysticks auch die Tastatur zur Eingabe genutzt werden. Ändern Sie hierzu bitte die Zeilen 1290 bis 1320 (Listing) entsprechend, und Sie können nun beispielsweise auch die Cursor-Tasten für die Steuerung der Schlange verwenden. Dabei steht M$ in der Zeile 1290 für »hoch«, in 1300 für »rechts«, in 1310 für »runter« und in 1320 für »links«.

(Alexander Jung/kn)

```
1000 REM +----------------------------+
1010 REM !                            !
1020 REM ! SNAKE        1986 BY A.JUNG !
1030 REM !                            !
1040 REM !  DATASETTENBENUTZUNG NACH  !
1050 REM !  JEDEM PROGRAMMSTART ERST  !
1060 REM !  NACH EINGABE VON:         !
1070 REM !                            !
1080 REM !   SYS 819 [CR]             !
1090 REM !                            !
1100 REM !  SONST SYSTEMABSTURZ !!!!  !
1110 REM !                            !
1120 REM +----------------------------+
1130 IF KN=0 THEN GOSUB 2340
1135 IF KN=1 THEN PRINT "{CLR,CTRL+N,WHITE}": COLOR 0,1: COLOR 4,1: GOSUB 2450
1140 :
1150 REM INITIALISIERUNG
1160 :
1170 COLOR 0,3,3: COLOR 1,8,7: COLOR 4,6,3
1180 DIM PO(DI),HI$(8,5): VOL 8: PU=0: PH=1: WO=5: RESTORE 2610
1190 FOR T1=0 TO 8: FOR T2=0 TO 5: READ HI$(T1,T2): IF HI$(T1,T2)="-1" THEN T2=6
1200 NEXT T2,T1: PU=-10
1210 FOR T=819 TO 871: READ A$: POKE T,DEC(A$): NEXT
1220 GOSUB 1780
1230 GOSUB 1970: PO=4012: PA=0: PE=0: L=5: R=-40 : V=R: PO(0)=PO: SP=0: POKE 239,0
1240 :
1250 REM HAUPTSCHLEIFE
1260 :
1270 POKE PO,215: POKE PO-1024,PEEK(65282): SOUND 3,1000,2
1280 V=R: GET A$
1290 IF A$="5" THEN R=-40
1300 IF A$="6" THEN R=1
1310 IF A$="R" THEN R=40
1320 IF A$="D" THEN R=-1
1330 IF ABS(R)>20 THEN POKE PO,221: ELSE POKE PO,192
1340 IF R=V THEN 1410
1350 IF R=-V THEN 1560
1360 IF SGN(R)<>SGN(V) THEN 1390
1370 IF R>V THEN POKE PO,238: ELSE POKE PO,237
1380 GOTO 1400
1390 IF SGN(R)=1 THEN POKE PO,240: ELSE POKE PO,253
1400 V=R
1410 PO=PO+R: PA=PA+1: IF PA>DI THEN PA=0
1420 IF PA=PE THEN POKE 3171,96: POKE PO(PE),32: PE=PA+1: SP=1: IF PE>DI THEN PE=0
1430 PO(PA)=PO: POKE PO-1024-V,102: IF PEEK(PO)<>32 THEN 1500
1440 IF L=>0 AND SP=0 THEN L=L-1: GOTO 1270
1450 POKE PO(PE),32: PE=PE+1: IF PE>DI THEN PE=0
1460 GOTO 1270
1470 :
1480 REM HINDERNISUEBERPRUEFUNG
1490 :
1500 HI=PEEK(PO)
1510 IF HI=81 THEN SOUND 3,400,6: GOSUB 1970: GOTO 1270
1520 IF HI=96 THEN 1700
1530 :
1540 REM SCHLANGE GEHT EIN
1550 :
1560 POKE PO,215: SOUND 3,800,3000
1570 FOR T=89 TO 0 STEP -1
1580 VOL T/10: POKE PO-1024,RND(1)*256
1590 NEXT : SOUND 3,0,0: Q2=.00001
1600 IF PA<PE THEN QQ=(DI-PE+PA): ELSE QQ=(PA-PE)
1610 QQ=QQ+.00001: FOR Q1=0 TO QQ: FOR T=1 TO 10 : NEXT
1620 SOUND 3,900,2: VOL 8: COLOR 4,6,8-Q2: COLOR 0,3,8-Q2: Q2=Q2+7.9/QQ
1630 POKE PO(PE),32: PE=PE+1: IF PE>DI THEN PE=0
1640 NEXT : FOR T=1 TO 100: NEXT : COLOR 0,3,3: COLOR 4,6,3
1650 WO=WO-1: IF WO=0 THEN 2090
1660 GOTO 1220
1670 :
1680 REM PHASE BEENDET
1690 :
1700 PU=PU+PH*10: WO=WO+1: PH=PH+1
1710 POKE PO,215: POKE PO-1024,247: RESTORE 2830 : VOL 8
1720 FOR T=1 TO 19: READ S1,S2
1730 COLOR 0,3,T/2.4: COLOR 4,6,T/2.4: SOUND 1,S1,S2: SOUND 1,1020,1
1740 NEXT : COLOR 0,3,3: COLOR 4,6,3: GOTO 1220
1750 :
1760 REM SPIELFELDMASKE
1770 :
1780 SYS 819: PRINT "{CLR,GREY3,RVSON}THE{3SPACE}SNAKE{4SPACE}*{4SPACE}THE{3SPACE}SNAKE{4SPACE}*{4SPACE}";
1790 PRINT USING "{HOME,DOWN,RVSON,YELLOW} PUNKTE : ##### ";PU;
1800 PRINT USING "{RVSON} LEBEN : ## ";WO;
1810 PRINT USING "{RVSON} PHASE : ##";PH;: POKE 3151,160: POKE 2127,119: SYS 830
1820 CHAR ,0,2,"{RVSON}########################################"
1830 FOR T=3 TO 23: CHAR ,0,T,"{RVSON}#": PRINT TAB(39)"#": NEXT
1840 CHAR ,0,24,"{RVSON}#######################################{LEFT,INST}#{HOME,RVOFF}": GOSUB 2040
1850 PG=PH: IF PG>10 THEN PG=INT(RND(1)*10+1)
1860 PG=PG-2: IF PH=1 THEN 1930
1870 FOR T=0 TO 5: IF HI$(PG,T)="-1" THEN 1930
1880 W1=VAL(LEFT$(HI$(PG,T),3))
1890 W2=VAL(MID$(HI$(PG,T),4,2))
1900 W3=VAL(RIGHT$(HI$(PG,T),2))
1910 FOR W=W1 TO W1+W2*W3 STEP W2
1920 POKE W+3072,163: NEXT W,T
1930 RETURN
1940 :
1950 REM APFEL DARSTELLEN
1960 :
1970 L=RND(1)*840+3192
1980 IF PEEK(L)<>32 THEN 1970
1990 POKE L,81: POKE L-1024,90: L=5: PU=PU+10
2000 SYS 819: PRINT USING "{HOME,DOWN,RVSON} PUNKTE : ##### ";PU;: SYS 830: RETURN
2010 :
2020 REM TASTATURABFRAGE
2030 :
2040 CHAR ,8,15,"DRUECKEN SIE EINE TASTE!": POKE 239,0: GET KEY S$
2050 CHAR ,8,15,"{24SPACE}": RETURN
2060 :
2070 REM BEERDIGUNG
2080 :
2090 SCNCLR : COLOR 0,4,0: COLOR 4,4,0: SYS 819
2100 PRINT TAB(15)"{RVOFF,WHITE}V": PRINT TAB(26)"C": PRINT TAB(39)"D": PRINT TAB(8)"V"
2110 PRINT "V" SPC(22)"F" SPC(8)"DI": PRINT "{LIG.RED,3SPACE}D{RVSON,2SPACE,RVOFF}T{RVSON}{RVOFF}F" SPC(22)"{WHITE}D{RVSON,2SPACE}C{RVOFF}"
2120 PRINT "{LIG.RED} D{RVSON,3SPACE,BROWN}B{LIG.RED,4SPACE,RVOFF,6SPACE,WHITE}F" SPC(14)"{RVSON}K{3SPACE}"
2130 PRINT "{LIG.RED} {RVSON,2SPACE,BROWN}B{LIG.RED}F{BROWN,RVOFF}B{LIG.RED}V{BROWN}K{RVSON,LIG.RED,3SPACE,RVOFF}F" SPC(20)"{WHITE,RVSON}F {RVOFF}V"
2140 PRINT "{LIG.RED,RVSON}V {BROWN}C{RVOFF}K{RVSON,LIG.RED} {BROWN,RVOFF}C{RVSON}D{RVOFF}B{RVSON,LIG.RED} {BROWN}B{LIG.RED,2SPACE}": PRINT "{RVSON}K C{BROWN,RVOFF}B{LIG.RED}C{RVSON} {BROWN,RVOFF}B{RVSON,LIG.RED}F{RVOFF,BROWN}TV{RVSON,LIG.RED,2SPACE}"
2150 PRINT "{RVSON,2SPACE,BROWN,RVOFF}BC{RVSON}D{LIG.RED}D{BROWN}VB{RVOFF}V{RVSON,LIG.RED,2SPACE}C" SPC(9)"{RVOFF,WHITE}DC" SPC(12)"D"
2160 PRINT "{LIG.RED}C{RVSON,2SPACE,BROWN}F{RVOFF}B{RVSON} {LIG.RED}VC{BROWN}B{LIG.RED,2SPACE,RVOFF}V" SPC(11)"{WHITE}D": PRINT "{2SPACE,LIG.RED,RVSON}F TI{BROWN} T{LIG.RED}FD"
2170 PRINT "{5SPACE,RVSON,BROWN,2SPACE}": PRINT "{4SPACE}D{RVSON,2SPACE,RVOFF,6SPACE,WHITE}F" SPC(20)"D": PRINT "{5SPACE,BROWN,RVSON,2SPACE}V{RVOFF}K"
```

Listing. Bitte beachten Sie bei der Eingabe von »Snake« die Hinweise zum Abtippen auf Seite 130

```
2180 PRINT "{5SPACE,RVSON,2SPACE}T": PRINT " {WH
     ITE}T{3SPACE,BROWN,RVSON,2SPACE,RVOFF}" SPC
     (17)"{WHITE}F": PRINT "{5SPACE,BROWN,RVSON,
     2SPACE,RVOFF}"
2190 PRINT "{5SPACE,BROWN,RVSON,2SPACE,RVOFF}T{4
     SPACE,ORANGE}T{RVSON,2SPACE,RVOFF}F" SPC(20
     )"{WHITE}T{RVSON}TC"
2200 PRINT "{4SPACE,BROWN,RVSON}T{3SPACE,4RIGHT,
     ORANGE,5SPACE,RVOFF,BROWN}TTT{CYAN,RVSON} {
     RVOFF}F" SPC(14)"{WHITE,RVSON}T{RVOFF}F{RVS
     ON}V"
2210 PRINT "{2SPACE,BROWN}TT{RVSON,4SPACE,RVOFF}
     T{GREEN}T{ORANGE}T{RVSON,7SPACE,RVOFF}T{GRE
     EN}T{RVSON,CYAN}T{GREEN} {RVOFF}T {BLACK} T
     TTTTTTF{GREEN} TT{WHITE,RVSON}TC {RVOFF,GRE
     EN}T"
2220 PRINT "{RVSON,GREEN,4SPACE,RVOFF}F{RVSON}V{
     RVOFF}F{RVSON}F{4SPACE,ORANGE,4SPACE,GREEN,
     7SPACE,RVOFF}F{BLACK}T{RVSON}RRRRRRRR{RVOFF
     }F{RVSON,GREEN,6SPACE}"
2230 PRINT "{RVSON,24SPACE,RVOFF}F{RVSON,BLACK}F
     {6SPACE}T{RVOFF,GREEN}T{RVSON,6SPACE}"
2240 PRINT "{RVSON,39SPACE,LEFT,INST} {HOME}"
2250 RESTORE 2900
2260 FOR T=22 TO 1 STEP -1: READ A,B
2270 SOUND 1,A,B: SOUND 1,1020,1: NEXT
2280 PRINT "{HOME}" CHR$(27)"DDRUECKEN SIE EINE
     TASTE FUER START !!{3SPACE}"
2290 PRINT USING "{RVSON}IHRE PUNKTE : ##### ";P
     U
2300 SYS 830: POKE 239,0: GET KEY S$: SYS 819: C
     LR : KN=1: GOTO 1130
2310 :
2320 REM TITEL
2330 :
2340 COLOR 0,1: COLOR 4,1
2350 RESTORE 3010
2360 FOR T=1 TO 8: READ A$: PRINT CHR$(27)A$
2370 FOR TT=1 TO 100: NEXT TT,T
2380 FOR T=1 TO 25: PRINT CHR$(27)"W{HOME}"
2390 FOR TT=1 TO 100: NEXT TT,T
2400 FOR T=1 TO 500: NEXT
2410 CHAR 1,0,24,CHR$(14): RESTORE 3120
2420 :
2430 REM SPIELREGELN
2440 :
2450 PRINT "SIND DIE SPIELREGELN BEKANNT ?{2SPAC
     E}(J/N)": POKE 239,0
2460 GET AN$: IF AN$="J" THEN RESTORE 3240: AN=7
     : GOTO 2480
2470 AN=19: IF AN$<>"N" THEN 2460
2480 FOR T=1 TO AN: VOL 8
2490 READ AN$: AN$=" "+AN$: FOR TT=1 TO LEN(AN$)
2500 PRINT CHR$(20) MID$(AN$,TT,1);"B";: SOUND 3
     ,1000,2
2510 FOR T1=1 TO 3: NEXT T1,TT
2520 PRINT CHR$(20): PRINT : FOR T1=1 TO 500: NE
     XT T1,T: POKE 239,0
2530 GET LA$: LA=VAL(LA$): IF LA>3 OR LA<1 THEN
     2530
2540 DI=50+20*LA
2550 FOR T=1 TO 25: PRINT CHR$(27)"W{HOME}"
2560 FOR TT=1 TO 100: NEXT TT,T: PRINT CHR$(142)
2570 RETURN
2580 :
2590 REM DATAS HINDERNISSE
2600 :
2610 DATA 5290121,-1
2620 DATA 5290121,3004006,579405,-1
2630 DATA 5290121,2894012,3104012,-1
2640 DATA 3004006,5394006,2894012,3104012,300011
     0,7690110
2650 DATA 1004006,7394006,5210106,5360107,552010
     6,-1
2660 DATA 3610106,3750109,3920106,6810106,695010
     9,7120106
2670 DATA 2934112,-1
2680 DATA 2934112,3053912,-1
2690 DATA 2934112,3084106,5254106,-1
2700 :
2710 REM DATAS MASCHINENROUTINE
2720 :
2730 DATA A9,0E,8D,14,03,A9,CE,8D
2740 DATA 15,03,60,A9,49,8D,14,03
2750 DATA A9,03,8D,15,03,60,18,A5
2760 DATA D8,69,10,85,D8,90,13,AE
2770 DATA 00,0C,A0,00,B9,01,0C,99
2780 DATA 00,0C,C8,C0,28,D0,F5,8E
2790 DATA 27,0C,4C,0E,CE
2800 :
2810 REM DATAS MUSIK
2820 :
2830 DATA 860,4,860,10,860,4
2840 DATA 900,14,900,14,912,14
2850 DATA 912,14,940,20,924,7
2860 DATA 900,7,1020,2,940,4
2870 DATA 942,10,900,4,876,14
2880 DATA 928,26,912,10,892,4
2890 DATA 900,26
2900 DATA 804,27,860,20,876,7
2910 DATA 892,27,900,20,892,7
2920 DATA 876,27,892,20,876,7
2930 DATA 860,27,804,13,1020,13
2940 DATA 804,27,860,20,876,7
2950 DATA 892,27,900,20,892,7
2960 DATA 876,27,892,20,876,7
2970 DATA 860,30
2980 :
2990 REM DATAS TITEL
3000 :
3010 DATA "X{2HOME,CLR,WHITE,3SPACE}¥{RVSON,3SPA
     CE,RVOFF}£{2SPACE,RVSON} {3RIGHT} {2RIGHT}
     {3RIGHT} {2RIGHT} {RVOFF,2SPACE}¥{RVSON} {R
     VOFF,2SPACE}¥{RVSON,4SPACE}"
3020 DATA "W{HOME,3SPACE,RVSON} ¥{RIGHT}£ {2RIGH
     T} {3RIGHT} {2RIGHT} {3RIGHT} {2RIGHT} {RVO
     FF} ¥{RVSON} ¥{2RIGHT} ¥"
3030 DATA "W{HOME,6SPACE}¥{RVSON} {2RIGHT} {2RIG
     HT,RVOFF}¥{RVSON} {2RIGHT} {RVOFF}£ ¥{RVSON
     } {2RIGHT} {RVOFF}¥{RVSON} ¥{3RIGHT} "
3040 DATA "W{HOME,5SPACE}¥{RVSON} ¥{2RIGHT} {RIG
     HT,RVOFF}¥{RVSON,2SPACE,2RIGHT,5SPACE,2RIGH
     T,2SPACE,RVOFF}£{4SPACE,RVSON,4SPACE}"
3050 DATA "W{HOME,3SPACE}¥{RVSON,2SPACE}¥{3RIGHT
     } {RVOFF}¥{RVSON} ¥ {2RIGHT} {3RIGHT} {2RIG
     HT} £ {RVOFF}£{3SPACE,RVSON} ¥"
3060 DATA "W{HOME,3SPACE,RVSON} ¥{5RIGHT,2SPACE}
     ¥{RIGHT} {2RIGHT} {RVOFF}£ ¥{RVSON} {2RIGHT
     } {RIGHT}£ {RVOFF}£{2SPACE,RVSON} "
3070 DATA "W{HOME,3SPACE,RVSON} {RVOFF}£ ¥{RVSON
     } {2RIGHT} ¥{2RIGHT} {2RIGHT}£{3SPACE}¥{2RI
     GHT} {2RIGHT}£ {2RIGHT} {RVOFF}£"
3080 DATA "W{HOME,3SPACE,RVSON}£{3SPACE}¥{2RIGHT
     } {3RIGHT} {3RIGHT}£ ¥{3RIGHT} {3RIGHT}£{2R
     IGHT}£{4SPACE}"
3090 :
3100 REM DATAS SPIELANLEITUNG
3110 :
3120 DATA "{5DOWN}"
3130 DATA "SPIELANLEITUNG:{25SPACE}============
     =="
3140 DATA "STEUERN SIE IHRE SCHLANGE MIT DEM{7SP
     ACE}JOYSTICK (PORT 1) !"
3150 DATA "FRESSEN SIE SOVIELE AEPFEL, WIE SIE{5
     SPACE}KOENNEN !"
3160 DATA "SIE WACHSEN STAENDIG. (KEIN GRUND ZUR
     {3SPACE}PANIK.)"
3170 DATA " NACH EINIGER ZEIT ERSCHEINT IN DER{6
     SPACE}OBEREN MAUER EIN AUSGANG."
3180 DATA "WENN SIE DURCH DIESEN DAS SPIELFELD V
     ER-LASSEN, ERHALTEN SIE EIN LEBEN MEHR."
3190 DATA "DER SCHWIERIGKEITSGRAD STEIGT MIT JED
     ER PHASE."
3200 DATA "DIE MAUER IST UNGENIESSBAR."
3210 DATA "SIE SELBST EBENFALLS."
3220 DATA "BEI JEDEM FEHLER GEHT DIE SCHLANGE EI
     N{2SPACE}UND VERLIERT EIN LEBEN."
3230 DATA "SIND ALLE LEBEN VERBRAUCHT, KOENNEN S
     IE IHRER BEERDIGUNG BEIWOHNEN."
3240 DATA ".....................................
     ..."
3250 DATA "{5DOWN}MADE BY ALEXANDER JUNG{5DOWN}"
3260 DATA "WAEHLEN SIE NUN DEN SCHWIERIGKEITSGRA
     D !{DOWN}"
3270 DATA " 1 = LEICHT"
3280 DATA " 2 = MITTEL"
3290 DATA " 3 = SCHWER"
3300 DATA "{9DOWN}"
```

Listing »Snake« (Schluß)

Die wilde Jagd durchs Labyrinth

Das hier abgedruckte Spiel »Puck« ist ein spannendes Geschicklichkeitsspiel, das an den Klassiker »Pacman« erinnert.

Puck ist ein in Basic programmiertes Spiel, das zwar einfach aussieht, aber ganz schön schwer zu spielen ist. Ein mit dem Joystick gesteuerter kleiner Puck muß durch ein Labyrinth laufen und dort alle Punkte fressen. Dabei wird er von einer gefährlichen Spinne verfolgt. Die genaue Spielanleitung befindet sich im Programm und wird beim Starten mit »RUN« angezeigt. (Karl-Eugen Laubacher/bs)

```
1 DIM F%(40,25)
2 REM DATA IN FELD F% LADEN ***
3 SCNCLR : GOSUB 271: SCNCLR : PRINT "EIN WENIG
  GEDULD BITTE!"
4 GOSUB 150
5 REM SPIELFELD ZEICHNEN ***
6 PU=5: V=0: Y=0: LV=25
7 IF LV<10 THEN LV=10
8 GOSUB 157: FOR N=1 TO 10
9 GET NR$: NEXT N
10 AD=88: H=0: NR=0: ZV=0
11 PW=38: PS=23: POKE 4030,81
12 GW=1: GS=1: POKE 3113,42: RE=32: OL=3
13 IF PS=GS AND PW=GW THEN GOSUB 178: IF PU=0 TH
   EN Y=Y+V: GOSUB 194: GOTO 6
14 ZU=INT(RND(1)*LV): IF ZU=0 THEN 17
15 GOSUB 19: IF V=541 THEN PL=5: PU=PU+1: Y=Y+54
   1: V=0: LV=LV-5: VOL 7: SOUND 1,800,15: GOTO
   7
16 IF PS=GS AND PW=GW THEN GOSUB 178: IF PU=0 TH
   EN Y=Y+V: GOSUB 194: GOTO 6
17 GOSUB 49
18 GOTO 13
19 NR=JOY(1)
20 IF AD=NR THEN GOSUB 29: H=0: RETURN
21 IF NR=0 THEN 25
22 ZV=AD: AD=NR: GN=0: GOSUB 29
23 IF GN=0 THEN H=0: RETURN
24 H=NR: AD=ZV: GOSUB 29: RETURN
25 IF H=0 THEN GOSUB 29: RETURN
26 ZV=AD: AD=H: GN=0: GOSUB 29
27 IF GN=0 THEN H=0: RETURN
28 AD=ZV: GOSUB 29: RETURN
29 IF AD<>7 THEN 34
30 C=3071+PW+40*PS: B=PEEK(C)
31 IF B=160 THEN GN=1: RETURN
32 IF B=46 THEN V=V+1
33 POKE C,81: PW=PW-1: POKE C+1,32: RETURN
34 IF AD<>3 THEN 39
35 C=3073+PW+PS*40: B=PEEK(C)
36 IF B=160 THEN GN=1: RETURN
37 IF B=46 THEN V=V+1
38 POKE C,81: PW=PW+1: POKE C-1,32: RETURN
39 IF AD<>1 THEN 44
40 C=3072+PW+40*PS-40: B=PEEK(C)
41 IF B=160 THEN GN=1: RETURN
42 IF B=46 THEN V=V+1
43 POKE C,81: PS=PS-1: POKE C+40,32: RETURN
44 IF AD<>5 THEN RETURN
45 C=3072+PW+40*PS+40: B=PEEK(C)
46 IF B=160 THEN GN=1: RETURN
47 IF B=46 THEN V=V+1
48 POKE C,81: PS=PS+1: POKE C-40,32: RETURN
49 GP%=F%(GW,GS)
50 A=3072+GW+40*GS
51 RA=RE
52 ON GP% GOTO 54,58,62,66,70,74,78,90,102,114,1
   26
53 REM 1 ***
54 IF OL=3 THEN 56
55 RE=PEEK(A+1): GW=GW+1: POKE A+1,42: POKE A,RA
   : RETURN
56 RE=PEEK(A-1): GW=GW-1: POKE A-1,42: POKE A,RA
   : RETURN
57 REM 2 ***
58 IF OL=4 THEN 60
59 RE=PEEK(A+40): GS=GS+1: POKE A+40,42: POKE A,
   RA: RETURN
60 RE=PEEK(A-40): GS=GS-1: POKE A-40,42: POKE A,
   RA: RETURN
61 REM 3 ***
62 IF OL=2 THEN 64
63 RE=PEEK(A-40): GS=GS-1: POKE A-40,42: OL=4: P
   OKE A,RA: RETURN
64 RE=PEEK(A+1): GW=GW+1: POKE A+1,42: OL=1: POK
   E A,RA: RETURN
65 REM 4 ***
66 IF OL=4 THEN 68
67 RE=PEEK(A+40): GS=GS+1: POKE A+40,42: OL=2: P
   OKE A,RA: RETURN
68 RE=PEEK(A+1): GW=GW+1: POKE A+1,42: OL=1: POK
   E A,RA: RETURN
69 REM 5 ***
70 IF OL=4 THEN 72
71 RE=PEEK(A+40): GS=GS+1: POKE A+40,42: OL=2: P
   OKE A,RA: RETURN
72 RE=PEEK(A-1): GW=GW-1: POKE A-1,42: OL=3: POK
   E A,RA: RETURN
73 REM 6 ***
74 IF OL=2 THEN 76
75 RE=PEEK(A-40): GS=GS-1: POKE A-40,42: OL=4: P
   OKE A,RA: RETURN
76 RE=PEEK(A-1): GW=GW-1: POKE A-1,42: OL=3: POK
   E A,RA: RETURN
77 REM 7 ***
78 IF OL<>2 THEN 81
79 AR=ABS(GW+1-PW)+ABS(GS-PS): AU=ABS(GW-PW)+ABS
   (GS+1-PS)
80 IF AR<AU THEN 88: ELSE 87
81 IF OL<>3 THEN 84
82 A1=ABS(GW-PW): AU=ABS(GS+1-PS)+A1: AO=ABS(GS-
   1-PS)+A1
83 IF AO<AU THEN 86: ELSE 87
84 AR=ABS(GW+1-PW)+ABS(GS-PS): AO=ABS(GW-PW)+ABS
   (GS-1-PS)
85 IF AR<AO THEN 88: ELSE 86
86 RE=PEEK(A-40): GS=GS-1: POKE A-40,42: OL=4: P
   OKE A,RA: RETURN
87 RE=PEEK(A+40): GS=GS+1: POKE A+40,42: OL=2: P
   OKE A,RA: RETURN
88 RE=PEEK(A+1): GW=GW+1: POKE A+1,42: OL=1: POK
   E A,RA: RETURN
89 REM 8 ***
90 IF OL<>1 THEN 93
91 AR=ABS(GW+1-PW)+ABS(GS-PS): AU=ABS(GW-PW)+ABS
   (GS+1-PS)
92 IF AR<AU THEN 98: ELSE 99
93 IF OL<>3 THEN 96
94 AL=ABS(GW-1-PW)+ABS(GS-PS): AU=ABS(GW-PW)+ABS
   (GS+1-PS)
95 IF AL<AU THEN 100: ELSE 99
96 A1=ABS(GS-PS): AL=ABS(GW-1-PW)+A1: AR=ABS(GW+
   1-PW)+A1
97 IF AL<AR THEN 100: ELSE 98
98 RE=PEEK(A+1): GW=GW+1: POKE A+1,42: OL=1: POK
   E A,RA: RETURN
99 RE=PEEK(A+40): GS=GS+1: POKE A+40,42: OL=2: P
   OKE A,RA: RETURN
100 RE=PEEK(A-1): GW=GW-1: POKE A-1,42: OL=3: PO
    KE A,RA: RETURN
101 REM 9 ***
102 IF OL<>1 THEN 105
103 A1=ABS(GW-PW): AU=ABS(GS+1-PS)+A1: AO=ABS(GS
    -1-PS)+A1
104 IF AO<AU THEN 110: ELSE 111
```

Listing 1. Das Listing zu »Puck« für den C16

```
105 IF OL<>2 THEN 108
106 AL=ABS(GW-1-PW)+ABS(GS-PS): AU=ABS(GW-PW)+AB
    S(GS+1-PS)
107 IF AL<AU THEN 112: ELSE 111
108 AL=ABS(GW-1-PW)+ABS(GS-PS): AO=ABS(GW-PW)+AB
    S(GS-1-PS)
109 IF AL<AO THEN 112: ELSE 110
110 RE=PEEK(A-40): GS=GS-1: POKE A-40,42: OL=4:
    POKE A,RA: RETURN
111 RE=PEEK(A+40): GS=GS+1: POKE A+40,42: OL=2:
    POKE A,RA: RETURN
112 RE=PEEK(A-1): GW=GW-1: POKE A-1,42: OL=3: PO
    KE A,RA: RETURN
113 REM 10 ***
114 IF OL<>1 THEN 117
115 AR=ABS(GW+1-PW)+ABS(GS-PS): AO=ABS(GW-PW)+AB
    S(GS-1-PS)
116 IF AR<AO THEN 123: ELSE 122
117 IF OL<>2 THEN 120
118 A1=ABS(GS-PS): AL=ABS(GW-1-PW)+A1: AR=ABS(GW
    +1-PW)+A1
119 IF AL<AR THEN 124: ELSE 123
120 AL=ABS(GW-1-PW)+ABS(GS-PS): AO=ABS(GW-PW)+AB
    S(GS-1-PS)
121 IF AL<AO THEN 124
122 RE=PEEK(A-40): GS=GS-1: POKE A-40,42: OL=4:
    POKE A,RA: RETURN
123 RE=PEEK(A+1): GW=GW+1: POKE A+1,42: OL=1: PO
    KE A,RA: RETURN
124 RE=PEEK(A-1): GW=GW-1: POKE A-1,42: OL=3: PO
    KE A,RA: RETURN
125 REM 11 ***
126 IF OL<>1 THEN 131
127 A1=ABS(GW-PW): AO=ABS(GS-1-PS)+A1: AU=ABS(GS
    +1-PS)+A1
128 AR=ABS(GW+1-PW)+ABS(GS-PS)
129 IF AO<AU AND AO<AR THEN 148
130 IF AR<AU THEN 145: ELSE 146
131 IF OL<>2 THEN 136
132 A1=ABS(GS-PS): AL=ABS(GW-1-PW)+A1: AR=ABS(GW
    +1-PW)+A1
133 AU=ABS(GS+1-PS)+ABS(GW-PW)
134 IF AU<AR AND AU<AL THEN 146
135 IF AL<AR THEN 147: ELSE 145
136 IF OL<>3 THEN 141
137 A1=ABS(GW-PW): AO=ABS(GS-1-PS)+A1: AU=ABS(GS
    +1-PS)+A1
138 AL=ABS(GW-1-PW)+ABS(GS-PS)
139 IF AO<AU AND AO<AL THEN 148
140 IF AL<AU THEN 147: ELSE 146
141 A1=ABS(GS-PS): AL=ABS(GW-1-PW)+A1: AR=ABS(GW
    +1-PW)+A1
142 AO=ABS(GW-PW)+ABS(GS-1-PS)
143 IF AO<AR AND AO<AL THEN 148
144 IF AL<AR THEN 147: ELSE 145
145 RE=PEEK(A+1): GW=GW+1: POKE A+1,42: OL=1: PO
    KE A,RA: RETURN
146 RE=PEEK(A+40): GS=GS+1: POKE A+40,42: OL=2:
    POKE A,RA: RETURN
147 RE=PEEK(A-1): GW=GW-1: POKE A-1,42: OL=3: PO
    KE A,RA: RETURN
148 RE=PEEK(A-40): GS=GS-1: POKE A-40,42: OL=4:
    POKE A,RA: RETURN
149 REM ENDE SPINNE BEWEGEN ***
150 RESTORE 218
151 FOR S=1 TO 23
152 FOR W=0 TO 39
153 READ A%: F%(W,S)=A%
154 NEXT W
155 NEXT S
156 RETURN
157 SCNCLR
158 RESTORE 264
159 FOR N=0 TO 19
160 READ A: POKE 16360+N,A: NEXT N
161 SYS 16360
162 FOR N=1 TO 40
163 POKE 3071+N,160
164 POKE 4031+N,160
165 NEXT N
166 POKE 3112,160
167 FOR N=0 TO 880 STEP 40
168 POKE 3151+N,160: POKE 3151+N+1,160
169 NEXT N
170 RESTORE 196
171 FOR N=0 TO 331
172 READ A%
173 POKE 3072+A%,160
174 NEXT N
175 T=PU: IF PU>9 THEN T=-15
176 POKE 4071,T+48
177 RETURN
178 VOL 7: VL=7
179 FOR SO=0 TO 30
180 SOUND 3,800,1: VOL VL: VL=VL-.23
181 NEXT SO
182 PU=PU-1
183 IF PU=0 THEN RETURN
184 IF RE=81 THEN RE=32
185 POKE 3072+PW+PS*40,RE
186 T=PU: IF PU>9 THEN T=-15
187 POKE 4071,T+48
188 PW=38: PS=23: POKE 4030,81
189 GW=1: GS=1: POKE 3113,42: RE=32: OL=3
190 FOR N=1 TO 10
191 GET NR$: NEXT N
192 AD=88: H=0: NR=0
193 RETURN
194 SCNCLR : PRINT "PUNKTE:";Y: PRINT : Y=0
195 INPUT "DRUECKE RETURN";DUMMY$: RETURN
196 DATA 82,84,85,86,87,89,90,91,93,94,95,97,99,
    100,101,102,103,105,106
197 DATA 107,109,111,112,114,116,117,122,127,135
    ,145,149,152,154,162
198 DATA 163,164,165,167,168,170,171,173,175,177
    ,179,180,181,183,185
199 DATA 187,188,189,190,192,194,195,197,208,213
    ,225,242,244,246,248
200 DATA 250,251,253,254,255,257,259,261,263,265
    ,267,268,269,271,273
201 DATA 275,276,277,284,286,288,290,297,299,301
    ,303,307,311,313,317
202 DATA 322,326,330,332,334,336,337,339,341,343
    ,345,346,347,349,351
203 DATA 353,354,355,362,364,366,367,369,370,372
    ,374,376,379,389,393
204 DATA 397,418,419,421,422,423,425,427,428,429
    ,431,433,435,442,443
205 DATA 444,445,446,448,449,450,451,452,454,456
    ,463,465,471,475,477
206 DATA 494,496,498,499,500,501,503,505,507,509
    ,510,511,513,514,515
207 DATA 517,522,523,524,526,527,528,530,532,533
    ,534,536,541,545,570
208 DATA 576,577,579,581,582,583,585,586,588,589
    ,590,591,592,594,595,596,597
209 DATA 602,604,605,606,607,608,612,614,617,619
    ,623,630,637,642,646
210 DATA 650,652,654,655,657,659,660,661,663,665
    ,667,668,670,672,673,674
211 DATA 675,682,684,686,688,689,690,701,703,705
    ,708,717,724,732,733,735,736
212 DATA 738,739,745,746,748,749,750,751,752,754
    ,755,757,762,763,764,766
213 DATA 767,768,770,773,778,781,783,792,794,797
    ,804,810,811,813,814
214 DATA 815,817,818,820,821,823,825,826,827,828
    ,829,830,834,836,837,842
215 DATA 844,846,848,853,857,860,863,872,874,877
    ,882,884,886,888,889
216 DATA 890,892,893,895,896,897,899,900,902,903
    ,904,905,907,908,909,911
217 DATA 912,914,915,917
218 DATA ,4,1,8,1,1,1,1,8,1,1,1,8,1,1,1,8,1,8,1
219 DATA 1,1,1,1,8,1,1,1,8,1,8,1,1,8,1,8,1,1,5,
220 DATA ,2,,2,,,,,2,,,,2,,,,2,,2,
221 DATA ,,,,2,,,,2,,2,,,2,,2,,,2,
222 DATA ,2,,3,1,1,5,,3,8,1,1,11,1,5,,7,1,11,1
223 DATA 1,1,8,1,9,,4,1,6,,3,5,,2,,3,8,1,9,
224 DATA ,2,,,,,2,,,2,,,2,,2,,2,,2,
225 DATA ,,2,,2,,2,,,,,2,,2,,,2,,2,
226 DATA ,7,1,8,1,8,10,5,,7,1,1,9,,3,1,11,1,11,1
227 DATA 8,1,11,1,9,,7,1,1,1,8,10,8,10,8,1,10,1,
    9,
228 DATA ,2,,2,,2,,2,,2,,,2,,,,2,,2,
229 DATA 2,,2,,2,,2,,,,2,,2,,2,,,,2,
230 DATA ,7,1,9,,2,,2,,2,,4,10,8,1,8,6,,2,
231 DATA 2,,2,,7,1,6,,4,1,9,,2,,3,1,5,,2,
232 DATA ,2,,7,1,9,,3,8,6,,2,,2,,2,,,2,
233 DATA 2,,2,,2,,,,2,,2,,2,,,,7,1,9,
234 DATA ,2,,2,,2,,,2,,,2,,2,,2,,4,6,
235 DATA 7,1,10,1,11,1,8,1,6,,7,1,9,,4,1,9,,2,
```

```
236 DATA ,7,1,10,1,10,1,8,10,1,1,10,1,11,1,11,1,
    9,,
237 DATA 2,,,,2,,2,,,,2,,2,,2,,7,1,9,
238 DATA ,2,,,,,,2,,,,,,2,,2,,7,1,1
239 DATA 10,1,5,,2,,7,1,8,1,6,,7,1,6,,2,,2,
240 DATA ,7,1,1,1,8,1,10,1,8,1,8,1,6,,2,,2,,
241 DATA ,,2,,2,,2,,2,,,,2,,,,2,,2,
242 DATA ,2,,,,2,,,,2,,2,,,,2,,3,8,1
243 DATA 5,,3,1,9,,3,8,10,1,1,1,10,8,1,1,10,1,9,
244 DATA ,7,1,8,1,10,1,1,1,9,,7,1,8,1,9,,,2,
245 DATA 2,,,,2,,,2,,,,,,2,,,,,2,
246 DATA ,2,,2,,,,,,7,1,9,,2,,3,5,,2,
247 DATA 3,1,5,,7,1,8,10,1,5,,4,1,10,1,1,5,,2,
248 DATA ,2,,7,1,5,,4,1,6,,2,,2,,,2,,2,
249 DATA ,,2,,2,,2,,,2,,2,,,,,7,1,9,
250 DATA ,2,,2,,2,,2,,,,7,1,10,8,1,10,8,10,1
251 DATA 5,,2,,2,,3,5,,3,1,10,1,8,1,1,9,,2,
252 DATA ,7,1,6,,7,1,10,1,8,1,9,,,2,,,2,,
253 DATA 7,1,11,1,9,,,2,,,,,,2,,,2,,2,
254 DATA ,2,,,,2,,,,2,,3,5,,3,1,8,6,,4
255 DATA 6,,2,,7,1,1,10,1,1,1,5,,2,,4,6,,2,
256 DATA ,7,1,5,,7,1,8,1,9,,,2,,,,2,,,2
257 DATA ,,2,,2,,,,,,7,1,9,,2,,,2,
258 DATA ,2,,2,,2,,2,,3,1,8,6,,4,1,6,,4,6
259 DATA ,4,6,,3,1,8,1,1,1,8,6,,2,,3,5,,2,
260 DATA ,2,,2,,2,,2,,,,2,,,2,,,,2,
261 DATA ,2,,,,,2,,,,2,,,2,,,2,,2,
262 DATA ,3,1,10,1,10,1,10,1,1,1,10,1,1,10,1,1,1
    ,10,1
263 DATA 1,10,1,1,1,1,10,1,1,1,10,1,1,10,1,1,10,
    1,6,
264 DATA 169,46,162,250,157,255,11,157,249,12,15
    7,243,13,157,237,14,202,208
265 DATA 241,96
270 VOL 7: SOUND 1,800,15
271 PRINT "ZIEL IST ES, MOEGLICHST VIELE PUNKTE
    ZU"
272 PRINT "FRESSEN, OHNE SICH VON DER SPINNE FAN
    -"
273 PRINT "GEN ZU LASSEN. DIE ANZAHL DER PUCKS"
274 PRINT "WIRD RECHTS UNTEN ANGEZEIGT."
275 PRINT "DER PUCK KANN MIT DEM JOYSTICK IN ALL
    E"
276 PRINT "RICHTUNGEN BEWEGT WERDEN. EINE GEWAEH
    LTE"
277 PRINT "RICHTUNG WIRD BEIBEHALTEN, SOLANGE DI
    ES"
278 PRINT "MOEGLICH IST. DER JOYSTICK KANN DABEI
     IN"
279 PRINT "NEUTRALSTELLUNG STEHEN."
280 PRINT "DURCH DAUERBETAETIGUNG ODER KURZES AN
    -"
281 PRINT "TIPPEN KANN EINE NEUE RICHTUNG GEWAEH
    LT"
282 PRINT "WERDEN, DIE BEI DER NAECHSTEN GELEGEN
    -"
283 PRINT "HEIT EINGESCHLAGEN WIRD."
284 PRINT "WURDE KEINE NEUE RICHTUNG GEWAEHLT,"
285 PRINT "BLEIBT DER PUCK STEHEN, WENN EINE BEW
    E-"
286 PRINT "GUNG IN DER ALTEN RICHTUNG NICHT MEHR
    "
287 PRINT "MOEGLICH IST."
290 PRINT "GELINGT ES ABZURAEUMEN, GIBT ES EINEN
    "
291 PRINT "PUCK ALS BONUS."
292 PRINT "BIS ZUR VIERTEN RUNDE WIRD DIE SPINNE
    "
293 PRINT "IM VERHAELTNIS ZUM PUCK VON RUNDE ZU"
294 PRINT "RUNDE ETWAS SCHNELLER."
295 PRINT "SIND ALLE PUCKS GEFANGEN, WIRD DIE ZA
    HL"
296 PRINT "DER GEFRESSENEN PUNKTE ANGEZEIGT."
297 INPUT "DRUECKE RETURN! ";D$
298 RETURN
299 END
```

Listing 1. Das Listing zu »Puck« für den C 16 (Schluß)

Cave – Sternenkampf im Labyrinth

Ihr Auftrag ist ein »Himmelfahrtskommando«. Doch vom Erfolg Ihres Unternehmens hängt das Überleben einer ganzen Flotte ab.

Auf dem Weg zurück zum Heimatplaneten Erde wird Ihr Geschwader durch feindliche Raumschiffe, Raketen und Minen bedroht (Bild). Die zivilen, unbewaffneten Raumschiffe folgen Ihrem Kampfjäger in sicherer Entfernung. Ihre Aufgabe ist es, ihnen mit drei Raumschiffen den Weg durch das feindliche Territorium zu bahnen. Besonders gefährlich sind die Raketen, die im Labyrinth auf Ihr Mutterschiff warten. Für die Zerstörung dieser stationierten Objekte gibt es deshalb auch die meisten Punkte. Feindliche Raumschiffe sind etwas leichter zu vernichten, deren Abschuß wird entsprechend geringer bewertet. Feindliche Minen, die sich häufig auch direkt in Ihrer Flugbahn befinden, dürfen auf keinen Fall abgeschossen werden, da die Detonation auch Ihr eigenes Raumschiff zerstören würde. Je tiefer Sie in das Höhlenlabyrinth eindringen, desto enger und schwerer wird es passierbar. Wenn es zum Manövrieren Ihres Schiffes zu eng wird, müssen Sie versuchen, sich einen Weg durch die Mauer zu schießen.

Zerstörte Raumschiffe und Raketen werden mit Pluspunkten belohnt. Wird Ihr Raumschiff jedoch durch Minen oder einen Zusammenprall zerstört, erhalten Sie einen Punkteabzug. Wieviel Punkte Ihnen abgezogen werden, hängt von der Anzahl bisher abgefeuerter Schüsse ab (Faktor 10). Wenn Sie beispielsweise bisher 30mal geschossen haben, verringert sich Ihr Punktekonto um 300. Wenn Sie bis zur Zerstörung Ihres letzten Raumschiffs genügend Punkte gesammelt haben, können Sie sich in die Liste der »Top Ten« eintragen. Finden Sie die optimale Strategie heraus.

(Alexander Jung/kn)

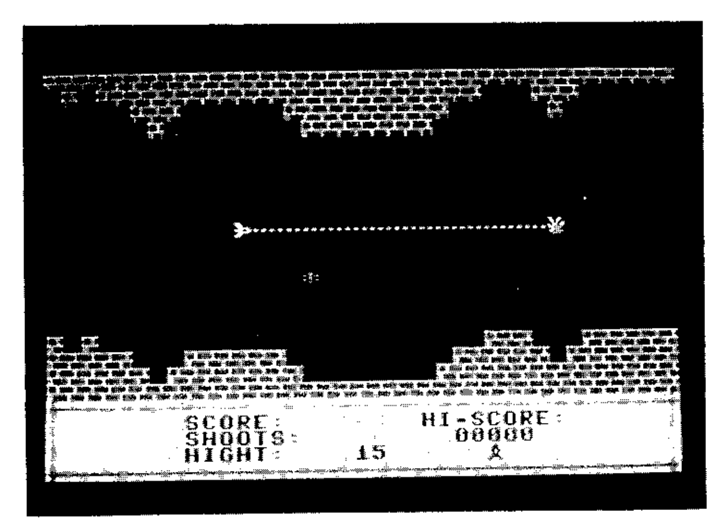
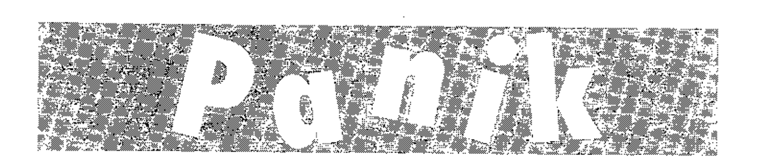


Bild. »Cave« besticht vor allem wegen seiner guten Grafik

```
1000 IF PEEK(0)<>15 THEN 1060
1010 POKE 55,255: POKE 56,55: CLR
1020 FOR T=15360 TO 15791
1030 READ A$: POKE T,DEC(A$)
1040 NEXT T: GOSUB 2280
1050 RESTORE 3890: FOR T=0 TO 9: READ HI$(T),HI(
     T): NEXT T
1060 PU=0: HI=HI(0): AN=0: SH=0: G=15
1070 POKE 65286,0: COLOR 0,1: COLOR 1,3,4: COLOR
      4,1: PRINT "{2HOME,CLR}": VOL 8: POKE 1343
     ,1
1080 CHAR ,0,20,"{RVSON,GREY3} [↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑
     ↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑[": PRINT
1090 PRINT "{RVSON,GREY3} ]{BLACK,7SPACE}SCORE:{
     8SPACE}HI-SCORE:{6SPACE,GREY3}]"
1100 PRINT "{RVSON} ]{WHITE} '()*{2SPACE,BLACK}S
     HOOTS:{9SPACE,RED}00000{WHITE,3SPACE}'()*{G
     REY3} ]"
1110 PRINT "{RVSON} ]{WHITE} +,./{2SPACE,BLACK}H
     IGHT:{18SPACE,WHITE}+,./{GREY3} ]"
1120 PRINT "{RVSON} [↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑
     ↑↑↑↑↑↑↑↑↑[{HOME}"
1130 CHAR ,16,21,"": PRINT USING "{BLUE,RVSON}##
     ###";PU
1140 CHAR ,18,22,"": PRINT USING "{BLUE,RVSON}##
     #";SH
1150 IF HI>9 THEN CHAR ,25,22,"": PRINT USING "{
     RVSON,RED}#####";HI
1160 CHAR ,25,23,"": FOR T=1 TO 3-AN: PRINT "{RV
     SON,YELLOW}";CHR$(34);CHR$(27);"O{RVSON} {R
     VOFF}";: NEXT T
1170 POKE 216,1: POKE 217,19
1180 FOR T=1 TO 40: SYS 15360: NEXT
1190 POKE 65286,27: SYS 15528
1200 IF AN=0 THEN CHAR ,7,8,"{YELLOW}ALEXANDER J
     UNG{2SPACE}PRESENTS :"
1210 FOR L=0 TO 1
1220 FOR T=0-7*L TO 7-7*L: COLOR 1,8,ABS(T)
1230 CHAR ,18,10,"'()*{DOWN,4LEFT}+,./"
1240 FOR P=1 TO 150: NEXT P,T
1250 FOR P=1 TO 500: NEXT P,L
1260 COLOR 1,3,4: CHAR ,1,8,CHR$(27)+"Q{2DOWN}"+
     CHR$(27)+"Q{DOWN}"+CHR$(27)+"Q"
1270 A=1: V=18: P=3473: PX=5: POKE 239,0: CHAR ,
     29-AN*2,23,"{RVSON,GREEN}"+CHR$(34)+CHR$(27
     )+"O"
1280 DO
1290 IF V>G THEN V=V-1: CHAR ,19,23,"": PRINT US
     ING "{RED,RVSON}##";V
1300 A=A+INT(RND(1)*3-1)
1310 IF A<1 THEN A=1
1320 IF A+V>19 THEN A=19-V
1330 POKE 216,A: POKE 217,A+V
1340 OP=P+PX
1350 GET J$: J=ASC(J$)
1360 P=P+(J=53)*40-(J=82)*40
1370 PX=ABS(PX-(J=54)+(J=68)): IF PX=37 THEN PX=
     36
1380 POKE OP,32: POKE OP-1024,66: SYS 15360
1390 IF PEEK(OP)<>32 OR PEEK(P+PX)<>32 THEN 1870
1400 POKE P+PX,31: POKE P+PX-1024,119
1410 IF INT(RND(1)*10)=3 THEN GOSUB 1650
1420 SOUND 3,-P+3833,7
1430 LOOP UNTIL J=84
1440 SH=SH+1: C=0: D=0: FOR T=P+PX TO P+37
1450 IF PEEK(T)<>32 AND PEEK(T)<>31 THEN D=PEEK(
     T): C=T-P-PX: T=P+37
1460 NEXT T: IF C=0 THEN C=37-PX
1470 C=C-1: F$=LEFT$("%%%%%%%%%%%%%%%%%%%%%%%%%%
     %%%%%%%%%%%",C)
1480 CHAR ,PX+2,(P-3073)/40,"{WHITE}"+F$
1490 IF D<>0 AND D<>35 THEN PRINT "{GREY1}&"
1500 SOUND 3,1000,10: SOUND 3,1000,1
1510 CHAR ,18,22,"": PRINT USING "{BLUE,RVSON}##
     #";SH
1520 FF$=LEFT$("{40SPACE}",C+1)
1530 CHAR ,PX+1,(P-3073)/40,"{YELLOW}←": COLOR 1
     ,3,4: PRINT FF$
1540 IF D=0 THEN 1280
1550 IF D=34 THEN PU=PU+100
1560 IF D=36 THEN PU=PU+50
1570 IF D<>35 THEN GOSUB 1800: GOTO 1280
1580 F$=LEFT$("GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG
     GGGGGGG",C)
1590 CHAR ,PX+1,(P-3073)/40,"{RVSON,YELLOW}¥{BLA
     CK}"+F$+"{CYAN}T"
1600 SOUND 3,800,999: FOR T=89 TO 0 STEP -1
1610 POKE 65304,RND(1)*256
1620 POKE 65305,RND(1)*256
1630 VOL(T/10): NEXT : SOUND 3,0,0: COLOR 4,1: V
     OL 8
1640 GOTO 1920
1650 IF RND(1)>.5 THEN 1690
1660 D=3110+A*40+INT(RND(1)*(V-1))*40
1670 POKE D,36: SOUND 1,300,10
1680 RETURN
1690 IF RND(1)>.5 THEN 1730
1700 D=3110+A*40+40*(V-1)
1710 POKE D,34: SOUND 1,500,10
1720 RETURN
1730 IF RND(1)>.5 THEN 1770
1740 D=3110+A*40+INT(RND(1)*2)*40*(V-1)
1750 POKE D,35: SOUND 1,700,10
1760 RETURN
1770 D=P+37
1780 POKE D,35: SOUND 1,900,10
1790 RETURN
1800 SOUND 3,800,999
1810 FOR T=159 TO 0 STEP -1
1820 VOL(T/20): NEXT
1830 SOUND 3,0,0: VOL 8
1840 CHAR ,16,21,"": PRINT USING "{BLUE,RVSON}##
     ###";PU
1850 IF SH/(5+G)=INT(SH/(5+G)) THEN G=G-1: IF G<
     3 THEN G=3
1860 RETURN
1870 P=P+PX: IF PEEK(OP)<>32 THEN P=OP
1880 POKE P,230
1890 SOUND 3,800,999: FOR T=89 TO 0 STEP -1
1900 POKE 65304,RND(1)*256
1910 VOL(T/10): NEXT : SOUND 3,0,0: COLOR 4,1: V
     OL 8
1920 AN=AN+1: PU=PU-SH*10
1930 IF AN<3 THEN 1070
1940 COLOR 0,2,5: COLOR 4,2,4: POKE 65286,27: PO
     KE 65287,8
1950 PRINT "{CLR,WHITE,2DOWN}";TAB(16)"{RVOFF}T{
     RVSON,6SPACE,RVOFF}T"
1960 PRINT TAB(16)"¥{RVSON,6SPACE}¥{RVOFF}"
1970 PRINT TAB(16)"{RVSON}T{6SPACE}T{RVOFF}"
1980 PRINT TAB(17)"E{RVSON,4SPACE,RVOFF}V"
1990 PRINT TAB(18)"E{RVSON,2SPACE,RVOFF}V"
2000 PRINT TAB(19)"{RVSON,2SPACE,RVOFF}"
2010 PRINT TAB(19)"{RVSON,2SPACE,RVOFF}"
2020 PRINT TAB(18)"B{RVSON,2SPACE,RVOFF}F"
2030 PRINT TAB(17)"B{RVSON,4SPACE,RVOFF}F"
2040 IF PU<HI(9) THEN 2200
2050 HI$="": A$="": PRINT CHR$(27)"T{4DOWN}": GO
     TO 2110
2060 GET A$: L=LEN(HI$)
2070 IF A$=CHR$(20) AND L>0 THEN HI$=LEFT$(HI$,L
     -1): GOTO 2110
2080 IF ASC(A$)>64 AND ASC(A$)<91 THEN HI$=HI$+A
     $
2090 IF A$=" " OR A$="-" THEN HI$=HI$+A$
2100 IF L=13 THEN A$=CHR$(20): GOTO 2070
2110 PRINT "{UP,2SPACE,BLACK}IHR NAME:{3SPACE}"H
     I$"← "
2120 IF A$<>CHR$(13) THEN 2060: ELSE SOUND 2,800
     ,10: SCNCLR
2130 IF LEN(HI$)<12 THEN HI$=HI$+" ": GOTO 2130
2140 HI$(10)=HI$: HI(10)=PU
2150 SP=0: FOR T=0 TO 9: IF HI$(T)="" THEN HI$(T
     )="{4SPACE}-----{4SPACE}"
2160 IF HI(T)>=HI(T+1) THEN 2190
2170 HI$=HI$(T): HI$(T)=HI$(T+1): HI$(T+1)=HI$
2180 HI=HI(T): HI(T)=HI(T+1): HI(T+1)=HI: SP=1
2190 NEXT T: IF SP=1 THEN 2150
2200 PRINT TAB(8)"{BLACK}NYYYRYYYYYYYYYYYYYRYYYYY
     M"
2210 FOR T=0 TO 9
2220 PRINT TAB(8)"H";: PRINT USING "## H";T+1;
2230 PRINT HI$(T)"H";: PRINT USING "#####H";HI(T
     )
2240 NEXT T
2250 PRINT TAB(8)"MPPPLPPPPPPPPPPPPPLPPPPPN"
2260 FOR T=1 TO 2000: NEXT
2270 POKE 239,0: GET KEY S$: GOTO 1060
2280 COLOR 0,10,3: COLOR 4,10,3: SCNCLR
2290 SYS 15528: POKE 65286,27
2300 FOR T=1 TO 25: A$="{BROWN,42SPACE}"
2310 FOR P=1 TO 3: MID$(A$,RND(1)*4-2+P*11,1)=MI
```

```
    D$("{RED,ORANGE,YELLOW}",P,1): NEXT P
2320 PRINT A$: NEXT T
2330 RESTORE 2990: VOL 8
2340 CHAR ,39,0,CHR$(27)+"T": A$=""
2350 FOR T=1 TO 90: FOR P=1 TO 60: NEXT
2360 PRINT "{HOME}";A$: SOUND 3,1010,2
2370 SYS 15411
2380 READ A$: IF LEFT$(A$,1)="-" THEN A$="{13DOW
    N}"+RIGHT$(A$,LEN(A$)-1)
2390 NEXT : A$="": POKE 239,0
2400 DO : GET A$
2410 SYS 15411: SOUND 3,1010,2
2420 FOR P=1 TO 60: NEXT P
2430 LOOP WHILE A$=""
2440 POKE 0,0: RETURN
2450 DATA A2,00,A9,27,8D,59,3C,A9
2460 DATA 0C,8D,5A,3C,A9,A1,8D,57
2470 DATA 3C,20,56,3C,E8,E4,D8,30
2480 DATA F8,A9,20,8D,57,3C,20,56
2490 DATA 3C,E8,E4,D9,30,F8,A9,A1
2500 DATA 8D,57,3C,20,56,3C,E8,E0
2510 DATA 14,30,F8,A9,00,85,DA,A9
2520 DATA 0C,85,DB,20,6A,3C,18,A5
2530 DATA DA,69,28,85,DA,B0,03,4C
2540 DATA 3B,3C,E6,DB,A5,DB,C9,0F
2550 DATA D0,01,60,4C,3B,3C,A9,A1
2560 DATA 8D,47,0F,18,AD,59,3C,69
2570 DATA 28,8D,59,3C,90,03,EE,5A
2580 DATA 3C,60,A5,DB,8D,92,3C,8D
2590 DATA 97,3C,8D,9A,3C,8D,A2,3C
2600 DATA 18,A5,DA,8D,91,3C,8D,99
2610 DATA 3C,69,01,8D,96,3C,69,26
2620 DATA 8D,A1,3C,90,03,EE,A2,3C
2630 DATA AE,F8,0E,A0,00,B9,F9,0E
2640 DATA 99,F8,0E,C8,C0,28,D0,F5
2650 DATA 8E,1F,0F,60,00,00,00,00
2660 DATA A2,00,BD,00,D0,9D,00,38
2670 DATA BD,00,D1,9D,00,39,BD,00
2680 DATA D2,9D,00,3A,BD,00,D3,9D
2690 DATA 00,3B,E8,E0,00,D0,E3,AD
2700 DATA 06,FF,09,40,8D,06,FF,AD
2710 DATA 07,FF,29,F7,8D,07,FF,AD
2720 DATA 12,FF,29,FB,8D,12,FF,AD
2730 DATA 13,FF,29,03,09,38,8D,13
2740 DATA FF,A9,72,8D,16,FF,A9,61
2750 DATA 8D,17,FF,A9,39,8D,18,FF
2760 DATA A2,00,BD,08,3D,9D,D8,38
2770 DATA E8,E0,A8,D0,F5,60,00,00
2780 DATA 18,18,18,FF,FF,18,18,18
2790 DATA 3C,42,99,A1,A1,99,42,3C
2800 DATA 18,18,18,18,18,18,18,18
2810 DATA 00,00,00,FF,FF,00,00,00
2820 DATA 00,E0,70,3E,F3,3E,70,E0
2830 DATA 00,00,00,00,00,00,00,00
2840 DATA 00,FB,FB,FB,00,DF,DF,DF
2850 DATA 08,1C,1C,36,14,1C,3E,63
2860 DATA 00,00,18,FF,18,FF,18,00
2870 DATA 00,18,7E,E7,7E,24,42,00
2880 DATA 00,00,00,00,66,00,00,00
2890 DATA C6,6C,29,BB,EE,3C,6F,14
2900 DATA 1E,33,60,60,60,61,33,1F
2910 DATA 00,00,00,00,F8,C9,6D,3F
2920 DATA 00,00,00,00,C7,ED,6C,38
2930 DATA 00,00,00,38,6C,F8,F0,3E
2940 DATA 00,3F,60,CC,DF,CD,61,3F
2950 DATA 00,C0,FC,0F,03,81,80,00
2960 DATA 00,00,00,3C,3C,00,00,00
2970 DATA 00,00,01,01,80,F0,3F,03
2980 DATA 00,FC,86,B3,FB,33,86,FC
2990 DATA "-←"
3000 DATA "-*"
3010 DATA "-*"
3020 DATA "-{RVSON} "
3030 DATA "-{RVSON}%"
3040 DATA "-{RVSON}#"
3050 DATA "-{RVSON}%"
3060 DATA "-{RVSON} "
3070 DATA "-{RVSON}C"
3080 DATA "-{RVSON}A"
3090 DATA "-{RVSON}V"
3100 DATA "-{RVSON}E"
3110 DATA "-{RVSON} "
3120 DATA "-{RVSON}%"
3130 DATA "-{RVSON}#"
3140 DATA "-{RVSON}%"
3150 DATA "-{RVSON} "
3160 DATA "-{RVSON}A"
3170 DATA "-{RVSON}L"
3180 DATA "-{RVSON}E"
3190 DATA "-{RVSON}X"
3200 DATA "-{RVSON}A"
3210 DATA "-{RVSON}N"
3220 DATA "-{RVSON}D"
3230 DATA "-{RVSON}E"
3240 DATA "-{RVSON}R"
3250 DATA "-{RVSON} "
3260 DATA "-{RVSON}J"
3270 DATA "-{RVSON}U"
3280 DATA "-{RVSON}N"
3290 DATA "-{RVSON}G"
3300 DATA "-{RVSON} "
3310 DATA "-{RVSON} "
3320 DATA "-{RVSON}1"
3330 DATA "-{RVSON}9"
3340 DATA "-{RVSON}8"
3350 DATA "-{RVSON}6"
3360 DATA "-{RVSON} "
3370 DATA "-{RVSON}%"
3380 DATA "-{RVSON}#"
3390 DATA "-{RVSON}%"
3400 DATA "-{RVSON} {RVOFF}"
3410 DATA "-V"
3420 DATA "-V"
3430 DATA "-V"
3440 DATA "{CLR,12DOWN}N M"
3450 DATA "{CLR,11DOWN}N{3SPACE}M"
3460 DATA "{10DOWN}N{5SPACE}M"
3470 DATA "{9DOWN}N{7SPACE}M"
3480 DATA "{9DOWN,RVSON,9SPACE}"
3490 DATA "{8DOWN,RVSON,9SPACE}T"
3500 DATA "{7DOWN,RVSON,10SPACE}"
3510 DATA "{6DOWN}T{RVSON,9SPACE}T"
3520 DATA "{6DOWN,RVSON,9SPACE}T"
3530 DATA "{6DOWN,RVSON,3SPACE}+'+{3SPACE}"
3540 DATA "{5DOWN}T{RVSON,3SPACE},(,{3SPACE}"
3550 DATA "{5DOWN,RVSON,4SPACE}.).{2SPACE}T"
3560 DATA "{5DOWN,RVSON,4SPACE}/*/{2SPACE}"
3570 DATA "{5DOWN,RVSON,9SPACE}T"
3580 DATA "{5DOWN}T{RVSON,9SPACE}"
3590 DATA "{6DOWN,RVSON,9SPACE}T"
3600 DATA "{6DOWN}T{RVSON,9SPACE}"
3610 DATA "{7DOWN,RVSON,9SPACE}"
3620 DATA "{7DOWN,RVSON,9SPACE}T"
3630 DATA "{7DOWN}T{RVSON,9SPACE}"
3640 DATA "{8DOWN,RVSON,9SPACE}T"
3650 DATA "{8DOWN}T{RVSON,9SPACE}"
3660 DATA "{9DOWN,RVSON,9SPACE}"
3670 DATA "{8DOWN}T{RVSON,9SPACE}"
3680 DATA "{8DOWN,RVSON,9SPACE}T"
3690 DATA "{8DOWN,RVSON,9SPACE}"
3700 DATA "{7DOWN}T{RVSON,9SPACE}"
3710 DATA "{7DOWN,RVSON,9SPACE}T"
3720 DATA "{7DOWN,RVSON,9SPACE}T"
3730 DATA "{7DOWN}T{RVSON,9SPACE}"
3740 DATA "{8DOWN,RVSON,9SPACE}"
3750 DATA "{8DOWN,RVSON,9SPACE}T"
3760 DATA "{8DOWN}T{RVSON,3SPACE}+'+{3SPACE}"
3770 DATA "{8DOWN}T{RVSON,3SPACE},(,{3SPACE}"
3780 DATA "{9DOWN,RVSON,3SPACE}.).{3SPACE}T"
3790 DATA "{9DOWN,RVSON,3SPACE}/*/{3SPACE}T"
3800 DATA "{9DOWN}T{RVSON,9SPACE}"
3810 DATA "{9DOWN}T{RVSON,9SPACE}"
3820 DATA "{10DOWN,RVSON,9SPACE}"
3830 DATA "{9DOWN}T{RVSON,9SPACE}"
3840 DATA "{9DOWN,RVSON,9SPACE}T"
3850 DATA "{9DOWN,RVSON,9SPACE}"
3860 DATA "{9DOWN}T{RVSON,8SPACE}T"
3870 DATA "{CLR,10DOWN,RVSON,8SPACE}T"
3880 DATA "{9DOWN,RVSON,9SPACE}T"
3890 DATA "------------",9
3900 DATA "{12SPACE}",8
3910 DATA "{4SPACE}+,./{4SPACE}",7
3920 DATA "{4SPACE}'()*{4SPACE}",6
3930 DATA "{4SPACE}+,./{4SPACE}",5
3940 DATA "{12SPACE}",4
3950 DATA "{4SPACE}MADE{4SPACE}",3
3960 DATA "{5SPACE}BY{5SPACE}",2
3970 DATA "{3SPACE}A JUNG{3SPACE}",1
3980 DATA " - £ 1986 - ",0
```

Listing. »Cave«. Bitte beachten Sie die Eingabehinweise auf Seite 130.



auf dem Bildschirm

»Apple Panic« ist ein flottes Geschicklichkeitsspiel für den C16, bei dem der Spieler nicht in Panik geraten darf. Beeilen Sie sich, aber behalten Sie die Nerven.

Das Spiel »Apple Panic« verlangt vom Spieler einiges an Nerven. Er muß Äpfel aufsammeln, ohne dabei in die Löcher im Fußboden zu fallen oder das Zeitlimit zu überschreiten.

Die Spielfigur wird mit den Cursortasten gesteuert. In der obersten Bildschirmzeile befinden sich drei Anzeigen: »Time« zeigt an, wie lange schon gespielt wird, »Score« gibt den Punktestand an und »E.Zeit« gibt die Zeit an, zu der man alle Äpfel eingesammelt haben muß. Die Anzeigen in der zweiten Zeile haben folgende Bedeutung: »Hi« gibt die höchste bisher erreichte Punktzahl an, »O« die Anzahl der noch verbleibenden Leben und »Level« die gerade durchlaufene Spielstufe.

Beim Abtippen und Laden des Programms muß man sehr sorgfältig arbeiten. Diskettenbesitzer gehen wie folgt vor (Hinweise für Datasettenbesitzer folgen im Anschluß): Zuerst geben Sie Listing 1 ein, welches die beiden anderen Programmteile nachlädt. Listing 1 wird als normales Basic-Programm eingegeben und mit »DSAVE "LOADER"« gespeichert.

Komplizierter wird es mit Listing 2, das den geänderten Zeichensatz enthält. Sie müssen unbedingt wie folgt vorgehen:

- Tippen Sie »MONITOR« und <RETURN> um in den Monitor zu gelangen.
- Dann geben Sie »T D000 D800 1000« und <RETURN> ein.
- Tippen Sie dann die Zeilen aus Listing 2 ein.
- Zum Speichern geben Sie »S "PRG.1 ",8,1000,1400« und <RETURN> ein.

Den dritten Programmteil geben Sie wie ein normales Basic-Programm ein und speichern es mit »DSAVE "APPLE PANIC"«.

Zum Starten geben Sie einfach »DLOAD "LOADER"« und nach dem Laden »RUN« ein.

Für Datasette:

Wenn Sie eine Datasette haben, brauchen Sie Listing 1 nicht abzutippen, haben dafür aber etwas mehr Arbeit, wenn Sie das Spiel laden wollen.

Zur Eingabe des Listing 2 gehen Sie wie oben beschrieben vor. Sie müssen allerdings mit »S "PRG.1 ",1,1000,1400« speichern! Das Listing 3 tippen Sie normal ein und speichern es mit »SAVE "APPLE PANIC"«.

Zum Laden von Datasette müssen Sie folgende Befehle eingeben:

»POKE 44,20:POKE 5120,0:NEW«
»LOAD "PRG.1 ",1,1«
»LOAD "APPLE PANIC"«
»RUN«

Dann sollte »Apple Panic« problemlos laufen.

Und nun viel Spaß und: Keine Panik!

(Andreas Zilla/bs)

```
10 REM LOADER
20 PRINT "{CLR}POKE44,20:POKE20*256,0:NEW"
25 PRINT "{3DOWN}LOAD"+CHR$(34)+"PRG.1"+CHR$(34)
   +",8,1"
30 PRINT "{4DOWN}LOAD"+CHR$(34)+"APPLE PANIC"+CH
   R$(34)+",8
40 PRINT "{4DOWN}RUN"
100 POKE 239,6: POKE 1319,19: POKE 1320,13: POKE
     1321,13: POKE 1322,13: POKE 1323,13: POKE 1
    324,13
```

Listing 1. Basic-Lader für »Apple Panic«. Bitte nur verwenden, wenn Sie ein Diskettenlaufwerk besitzen.

```
>1000 04 08 10 7e 7e 3c 18 00 :
>1008 18 10 fc f0 f0 f8 7e 3c :
>1010 3c 7f ff ff e9 c7 42 00 :
>1018 21 10 3f 0f 0f 1f 7f 3c :
>1020 00 30 10 e5 ff ff 72 36 :
>1028 7e 60 60 78 60 60 7e 00 :
>1030 38 10 60 60 60 66 3c 00 :
>1038 40 10 60 6e 66 66 3c 00 :
>1040 48 10 66 7e 66 66 66 00 :
>1048 50 10 18 18 18 18 3c 00 :
```

Listing 2. Maschinensprache-Teil von »Apple Panic«. Bitte unbedingt die Eingabehinweise im Text beachten!

```
10 REM **********************************
20 REM * A P P L E        P A N I C  1 *
30 REM *                               *
40 REM *      VON   ANDREAS ZILLA      *
50 REM *                               *
60 REM **********************************
100 GOSUB 2040: REM TITELBILD
110 REM HAUPTPROGRAMMM
120 GET JJ$: IF JJ$="" THEN 120
130 SC=0: BI=0: LE=3
140 BS=3072: X=0: Y=3: PU=14: TI$="000000": A=1
145 AA$(1)="000230": AA$(2)="000210": AA$(3)="00
    0200"
150 IF RO=1 THEN AA$(1)="000210": AA$(2)="000150
    ": AA$(3)="000120"
160 IF RO=2 THEN AA$(1)="000150": AA$(2)="000120
    ": AA$(3)="000100"
170 IF RO=3 THEN AA$(1)="000150": AA$(2)="000120
    ": AA$(3)="000050"
180 AA$(4)=AA$(1): AA$(5)=AA$(2)
190 POKE 65298,192: POKE 65299,17
200 BI=BI+1: IF BI>5 THEN BI=1: RO=RO+1: RESTORE

210 IF BI=1 THEN GOSUB 720
220 IF BI=2 THEN GOSUB 990
```

Listing 3. Hauptprogramm zu »Apple Panic«

```
230 IF BI=3 THEN GOSUB 1260
240 IF BI=4 THEN GOSUB 1520
250 IF BI=5 THEN GOSUB 1780
260 PRINT "{HOME,22RIGHT}E.ZEIT" MID$(AA$(BI),4,
    3)
290 :
300 :
310 POKE BS+X+40*Y,A: XA=X: YA=Y: GET A$
320 PRINT "{HOME}TIME:" RIGHT$(TI$,3);" SFORE:";
    SC
330 PRINT "HI=";HI;"{5SPACE}A=";LE;"{6SPACE}LEVE
    L:";LV
340 IF RO=0 THEN LV=BI
350 IF RO=1 THEN LV=BI+5
360 IF RO=2 THEN LV=BI+10
370 IF LE=0 THEN 2240
380 IF A$="{UP}" THEN A=4
390 IF A$="{RIGHT}" THEN A=1
400 IF A$="{DOWN}" THEN A=2
410 IF A$="{LEFT}" THEN A=3
420 IF A=1 THEN DY=0: DX=1
430 IF A=2 THEN DY=1: DX=0
440 IF A=3 THEN DX=-1: DY=0
450 IF A=4 THEN DX=0: DY=-1
460 Q=PEEK(BS+X+DX+40*(Y+DY))
470 IF Q<>32 THEN 570
480 X=X+DX: Y=Y+DY
490 IF PU=0 THEN 140
500 IF LE=0 THEN 2240
510 VOL 8: SOUND 3,100,2
520 IF BI=1 AND TI$>AA$(1) THEN 2240
530 IF BI=2 AND TI$>AA$(2) THEN 2240
540 IF BI=3 AND TI$>AA$(3) THEN 2240
542 IF BI=4 AND TI$>AA$(4) THEN 2240
544 IF BI=5 AND TI$>AA$(5) THEN 2240
550 POKE BS+XA+40*YA,32
560 GOTO 310
570 IF Q=102 THEN 310
580 IF Q=0 THEN 670
590 IF Q=87 THEN 610
600 GOTO 480
610 REM LOCH
620 X=X+DX: Y=Y+DY: POKE BS+XA+40*YA,32
630 POKE BS+X+40*Y,42: VOL 8: SOUND 1,100,50: FO
    R I=1 TO 1000: NEXT
640 POKE BS+X+40*Y,87
650 X=0: Y=3: DX=0: DY=0: LE=LE-1
660 GOTO 310
670 REM APFEL
680 X=X+DX: Y=Y+DY: POKE BS+XA+40*YA,32: VOL 8:
    SOUND 1,800,30: PU=PU-1
690 POKE BS+X+40*Y,0: SC=SC+100
700 GOTO 310
710 REM BILD1*************************
720 PRINT "{CLR,4DOWN}₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸
    ₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸";
730 PRINT "{39SPACE}₸";
740 PRINT "₸₸ ₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸ ₸₸₸
    ₸₸₸";
750 PRINT "₸₸{2SPACE}₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸
    ₸₸ ₸₸₸₸₸₸";
760 PRINT "₸₸₸{36SPACE}₸";
770 PRINT "₸₸₸ ₸₸₸ ₸₸₸₸₸₸₸₸₸₸ ₸₸₸₸₸₸₸₸₸₸ ₸₸₸₸₸₸₸
    ₸₸₸";
780 PRINT "₸₸₸{20SPACE}₸₸₸₸₸₸{2SPACE}₸₸{6SPACE}₸
    ";
790 PRINT "₸₸₸₸₸ ₸₸₸₸ ₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸{6SPACE}
    ₸₸₸₸₸";
800 PRINT "₸₸₸₸₸{26SPACE}₸₸₸ ₸₸₸₸₸";
810 PRINT "₸₸₸{3SPACE}₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸ ₸
    ₸₸{5SPACE}₸";
820 PRINT "₸₸₸ ₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸{2SPACE
    }₸₸ ₸₸₸₸₸";
830 PRINT "₸₸{37SPACE}₸";
840 PRINT "₸₸ ₸₸₸{2SPACE}₸₸₸₸₸ ₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸
    ₸₸₸₸₸₸₸₸₸₸";
850 PRINT "₸₸ ₸₸₸₸{2SPACE}₸₸₸₸ ₸₸₸₸₸₸₸₸₸{15SPACE
    }₸₸";
860 PRINT "₸₸ ₸₸₸₸{SHF.SP,2SPACE}₸₸₸ ₸₸{8SPACE}₸
    ₸₸₸₸₸₸₸₸ ₸₸₸₸₸₸";
870 PRINT "₸₸ ₸₸₸₸₸₸ ₸₸₸ ₸₸ ₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸ ₸₸₸
    ₸₸₸";
880 PRINT "₸₸{34SPACE}₸₸₸₸";
890 PRINT "₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸ ₸₸₸₸₸₸₸₸₸₸₸₸ ₸
    ₸₸₸";
900 PRINT "₸₸{31SPACE}₸₸ ₸₸₸₸";
910 PRINT "₸₸ ₸₸₸₸ ₸₸₸₸₸₸ ₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸ ₸₸ ₸
    ₸₸₸";
920 PRINT "₸₸{34SPACE}₸₸₸₸";
930 PRINT "₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸
    ₸₸₸";
940 DATA 3275,3201,3215,3230,3345,3582,3549,3610
    ,3819,3901,3973,3714,3816,3412
950 FOR I=1 TO 14: READ BD: POKE BD,0: NEXT
960 DATA 3209,3347,3324,3355,3608,3719,3815,3974
970 FOR I=1 TO 8: READ BD: POKE BD,87: NEXT : RE
    TURN
980 REM BILD 2 ***********************
990 PRINT "{CLR,5DOWN}₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸
    ₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸";
1000 PRINT "₸{38SPACE}₸";
1010 PRINT "₸ ₸₸₸₸₸₸₸₸₸ ₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸
     ₸₸ ₸";
1020 PRINT "₸ ₸₸₸₸₸₸₸₸₸ ₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸
     ₸₸ ₸";
1030 PRINT "₸ ₸₸₸{30SPACE}₸₸₸ ₸";
1040 PRINT "₸ ₸₸₸ ₸₸₸₸₸₸₸ ₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸ ₸
     ₸₸ ₸";
1050 PRINT "₸ ₸₸₸ ₸₸₸₸₸₸₸ ₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸{5
     SPACE}₸";
1060 PRINT "₸ ₸₸₸ ₸₸₸{22SPACE}₸₸₸ ₸₸₸ ₸";
1070 PRINT "₸ ₸₸₸{5SPACE}₸₸₸₸₸₸₸₸₸ ₸₸₸₸₸₸₸₸₸₸₸ ₸₸
     ₸ ₸₸₸ ₸";
1080 PRINT "₸ ₸₸₸ ₸₸₸ ₸₸₸₸₸₸₸₸{10SPACE}₸₸ ₸₸₸ ₸₸
     ₸ ₸";
1090 PRINT "₸ ₸₸₸ ₸₸₸ ₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸{3SPACE}₸
     ₸₸₸ ₸₸₸ ₸";
1100 PRINT "₸ ₸₸₸ ₸₸₸ ₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸ ₸ ₸₸₸ ₸
     ₸₸ ₸";
1110 PRINT "₸ ₸₸₸ ₸₸₸{22SPACE}₸₸₸ ₸₸₸ ₸";
1120 PRINT "₸ ₸₸₸ ₸₸₸ ₸₸₸₸ ₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸ ₸₸₸ ₸
     ₸₸ ₸";
1130 PRINT "₸ ₸₸₸ ₸₸₸ ₸₸₸₸{13SPACE}₸₸₸ ₸₸₸{5SPAC
     E}₸";
1140 PRINT "₸ ₸₸₸ ₸₸₸ ₸₸₸₸₸₸₸₸₸₸₸ ₸₸₸₸₸₸₸₸ ₸₸₸ ₸
     ₸₸ ₸";
1150 PRINT "₸ ₸₸₸ ₸₸₸{22SPACE}₸₸₸ ₸₸₸ ₸";
1160 PRINT "₸ ₸₸₸ ₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸ ₸₸₸₸ ₸
     ₸₸ ₸";
1170 PRINT "₸ ₸₸₸{30SPACE}₸₸₸ ₸";
1180 PRINT "₸ ₸₸₸ ₸₸₸₸₸₸₸₸₸ ₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸
     ₸₸ ₸";
1190 PRINT "₸{38SPACE}₸";
1200 PRINT "₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸
     ₸₸₸₸";
1210 DATA 3197,3214,3228,3630,3906,3894,3960,363
     7,3681,3729,3538,3513,3817,3532
1220 FOR I=1 TO 14: READ BD: POKE BD,0: NEXT
1230 DATA 3753,3717,3809,3578,3334,3626,3470,396
     1
1240 FOR I=1 TO 8: READ BD: POKE BD,87: NEXT : R
     ETURN
1250 REM ***************************
1260 PRINT "{CLR,5DOWN}₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸
     ₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸";
1270 PRINT "{15SPACE}₸{19SPACE}₸₸₸₸₸";
1280 PRINT "₸₸₸₸₸₸₸₸₸₸₸₸{3SPACE}₸ ₸₸₸₸₸₸₸₸₸₸₸₸₸₸
     ₸₸₸{2SPACE}₸₸₸₸";
1290 PRINT "₸{10SPACE}₸₸₸ ₸{22SPACE}₸₸";
1300 PRINT "₸{2SPACE}₸₸₸₸₸₸₸₸₸{2SPACE}₸₸₸₸₸₸₸ ₸
     ₸₸₸₸₸₸₸₸₸₸₸₸ ₸₸₸₸";
1310 PRINT "₸{13SPACE}₸₸₸₸₸₸₸₸ ₸₸₸₸₸₸₸{6SPACE}₸₸
     ₸₸";
1320 PRINT "₸ ₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸₸{6SPACE}₸₸₸₸₸₸
     ₸{4SPACE}₸";
1330 PRINT "₸ ₸{14SPACE}₸₸₸₸₸₸₸₸₸₸ ₸₸₸₸₸₸₸₸₸₸₸₸"
     ;
1340 PRINT "₸ ₸₸₸ ₸₸₸ ₸₸ ₸₸₸{22SPACE}₸₸";
```

Listing 3. Hauptprogramm zu »Apple Panic«. Bitte unbedingt die Eingabehinweise im Text beachten.

```
1350 PRINT "╤ ╤╤╤ ╤╤╤{4SPACE}╤╤╤ ╤╤╤╤╤╤{4SPACE}╤
     ╤╤╤ ╤╤╤╤╤╤╤╤";
1360 PRINT "╤ ╤╤╤ ╤╤╤╤ ╤╤╤╤╤{10SPACE}╤╤╤╤╤ ╤╤╤╤
     ╤╤╤";
1370 PRINT "╤ ╤╤╤╤╤╤╤╤╤╤╤╤╤ ╤╤╤╤╤╤{2SPACE}╤╤╤╤╤
     ╤ ╤╤╤╤ ╤╤╤";
1380 PRINT "╤{10SPACE}╤╤╤╤╤{8SPACE}╤╤╤{10SPACE}╤
     ╤╤";
1390 PRINT "╤ ╤╤╤ ╤╤╤╤ ╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤ ╤╤╤╤
     ╤╤╤╤";
1400 PRINT "╤ ╤╤╤ ╤╤╤╤{28SPACE}╤╤";
1410 PRINT "╤ ╤╤╤{4SPACE}╤╤╤{6SPACE}╤╤╤╤╤╤╤ ╤╤╤╤
     ╤╤ ╤╤╤╤╤╤╤";
1420 PRINT "╤{3SPACE}╤╤╤╤ ╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤ ╤╤╤╤╤
     ╤ ╤╤╤╤╤╤╤";
1430 PRINT "╤╤╤ ╤╤╤╤{30SPACE}╤╤";
1440 PRINT "╤╤╤ ╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤ ╤╤╤╤╤╤╤╤╤╤╤╤╤╤
     ╤ ╤╤";
1450 PRINT "╤╤╤{14SPACE}╤╤╤╤ ╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤ ╤╤"
     ;
1460 PRINT "╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤{22SPACE}╤╤";
1470 PRINT "╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤
     ╤╤╤╤";
1480 DATA 3194,3326,3357,3279,3673,3924,3853,374
     9,3570,3490,3651,3303,3448,3557
1490 FOR I=1 TO 14: READ BD: POKE BD,0: NEXT
1500 DATA 3245,3314,3713,3860,3537,3577,3304,352
     1,3562,3436
1510 FOR I=1 TO 10: READ BD: POKE BD,87: NEXT :
     RETURN
1520 REM BILD 4**************************
1530 PRINT "{CLR,4DOWN}╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤
     ╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤";
1540 PRINT "{14SPACE}╤{2SPACE}╤ ╤{2SPACE}╤{2SPAC
     E}╤╤╤╤╤╤╤╤╤╤╤{3SPACE}╤";
1550 PRINT "╤{3SPACE}╤╤╤╤╤╤╤{3SPACE}╤{20SPACE}╤╤
     ╤╤╤";
1560 PRINT "╤{3SPACE}╤╤{6SPACE}╤ ╤╤╤╤╤╤╤╤╤ ╤╤╤╤╤
     ╤╤ ╤H̲{5SPACE}╤";
1570 PRINT "╤ H̲{7SPACE}╤{3SPACE}╤{2SPACE}H̲{3SPAC
     E}H̲{5SPACE}╤╤╤╤ ╤╤╤╤╤╤╤╤";
1580 PRINT "╤╤╤╤╤╤╤╤╤{3SPACE}H̲ ╤╤{3SPACE}╤╤╤╤{2S
     PACE}╤╤╤H̲ ╤{2SPACE}╤ ╤ ╤ ╤";
1590 PRINT "╤╤╤╤╤╤╤╤╤╤╤ ╤╤╤{2SPACE}╤╤╤╤╤╤╤ H̲╤╤{2
     SPACE}╤{8SPACE}╤";
1600 PRINT "╤{11SPACE}╤{4SPACE}╤{9SPACE}╤ ╤╤╤╤╤╤
     ╤╤╤ ╤";
1610 PRINT "╤ ╤╤╤╤╤╤╤╤╤╤{2SPACE}H̲ ╤╤╤╤╤╤╤╤╤ ╤ ╤
     {5SPACE}╤╤╤ ╤";
1620 PRINT "╤{16SPACE}╤╤╤╤╤╤{4SPACE}╤{4SPACE}╤╤{
     5SPACE}╤";
1630 PRINT "╤╤╤╤╤╤╤ ╤╤╤╤╤╤╤ ╤╤╤╤╤╤╤ ╤╤╤╤╤╤╤ ╤╤ ╤
     ╤╤╤╤";
1640 PRINT "╤{5SPACE}╤{32SPACE}╤";
1650 PRINT "╤ ╤╤╤╤╤ ╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤ ╤╤╤╤╤
     ╤ ╤╤";
1660 PRINT "╤ ╤╤╤╤╤ H̲╤╤╤╤╤╤╤╤ ╤╤╤╤╤╤╤ ╤╤╤╤ ╤╤╤╤╤
     ╤ ╤╤";
1670 PRINT "╤{38SPACE}╤";
1680 PRINT "╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤ ╤╤╤╤ ╤╤╤╤╤╤╤╤
     ╤╤╤╤";
1690 PRINT "╤{20SPACE}╤ ╤╤╤╤{12SPACE}╤";
1700 PRINT "╤ ╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤ ╤╤╤╤╤╤╤╤╤╤╤╤╤╤
     ╤╤ ╤";
1710 PRINT "╤{13SPACE}╤╤╤╤ H̲{15SPACE}╤╤╤ ╤";
1720 PRINT "╤{2SPACE}╤╤╤╤╤╤╤╤╤{9SPACE}H̲{2SPACE}╤
     ╤╤╤╤╤╤╤╤╤{5SPACE}╤";
1730 PRINT "╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤{3SPACE}╤╤╤╤╤╤╤╤
     ╤╤╤╤╤╤╤╤╤";
1740 PRINT "╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤
     ╤╤╤╤";
1750 FOR I=1 TO 14: READ BD: POKE BD,0: NEXT
1760 DATA 3238,3405,3318,3473,3446,3503,3428,335
     0,3250,3415,3658,3936,3964,3986
1770 RETURN
1780 REM BILD 5 **************************
1790 PRINT "{CLR,5DOWN}╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤
     ╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤";
1800 PRINT "{13SPACE}╤╤ H̲╤╤╤╤╤╤╤H̲{2SPACE}╤╤╤╤╤╤╤
     ╤╤╤╤╤╤";
1810 PRINT "╤╤ ╤╤╤╤╤╤╤╤╤{27SPACE}╤";
1820 PRINT "╤╤{6SPACE}H̲{4SPACE}╤╤╤╤╤ ╤╤╤╤ ╤╤╤╤╤╤
     ╤ ╤╤╤ ╤╤ ╤";
1830 PRINT "╤╤ ╤╤╤ ╤╤╤╤╤ ╤╤╤╤╤{2SPACE}╤╤╤{6SPACE
     }╤╤ ╤╤╤ ╤╤ ╤";
1840 PRINT "╤╤ ╤╤╤ ╤╤╤╤╤{2SPACE}H̲╤╤╤╤ ╤╤╤ ╤╤╤╤{4
     SPACE}╤╤╤ ╤╤ ╤";
1850 PRINT "╤╤ ╤╤╤{18SPACE}╤ ╤╤ ╤╤{2SPACE}H̲{2SPA
     CE}╤╤ ╤";
1860 PRINT "╤╤ ╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤ ╤╤ ╤╤╤╤╤╤╤
     ╤╤ ╤";
1870 PRINT "╤╤{24SPACE}╤╤{11SPACE}╤";
1880 PRINT "╤╤╤╤ ╤╤╤╤╤ ╤╤╤╤╤ ╤╤╤ ╤╤╤{3SPACE}╤╤╤╤
     ╤╤╤ ╤╤╤╤╤";
1890 PRINT "╤╤{3SPACE}╤╤╤╤╤{4SPACE}╤╤ ╤╤╤{5SPACE
     }H̲ ╤╤╤╤╤╤╤{4SPACE}╤╤";
1900 PRINT "╤╤ ╤╤╤╤╤╤╤╤╤╤ ╤╤ ╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤ ╤
     ╤ ╤╤";
1910 PRINT "╤╤ ╤H̲{5SPACE}╤╤╤ ╤╤{19SPACE}╤╤ ╤╤";
1920 PRINT "╤╤{4SPACE}╤╤╤{5SPACE}╤╤╤╤╤{2SPACE}╤╤
     ╤╤╤ ╤╤╤╤╤╤╤╤╤╤ ╤╤";
1930 PRINT "╤╤╤╤ ╤╤╤╤╤╤╤╤ ╤╤╤╤H̲{2SPACE}╤╤╤╤╤{12S
     PACE}╤╤";
1940 PRINT "╤╤{25SPACE}╤╤╤╤╤╤╤ ╤╤╤╤╤";
1950 PRINT "╤╤ ╤╤╤╤╤╤╤╤╤╤╤╤ ╤╤╤╤╤╤╤╤ ╤╤╤╤╤╤╤╤╤ ╤
     ╤╤╤╤";
1960 PRINT "╤╤{33SPACE}╤╤╤╤╤";
1970 PRINT "╤╤╤╤ ╤╤╤╤╤╤╤ ╤╤╤╤╤╤ ╤╤╤╤╤╤╤╤╤╤╤╤╤╤ ╤
     ╤╤╤╤";
1980 PRINT "╤╤╤╤{9SPACE}╤╤╤╤╤╤{16SPACE}╤╤╤╤╤";
1990 PRINT "╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤
     ╤╤╤╤";
2000 PRINT "╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤╤
     ╤╤╤╤";
2010 FOR I=1 TO 14: READ BD: POKE BD,0: NEXT
2020 DATA 3370,3438,3677,3874,3962,3979,3789,354
     6,3466,3258,3449,3529,3578,3612
2030 RETURN
2040 REM TITELBILD
2050 PRINT "{CLR}";: POKE 65298,196: POKE 65299,
     208
2060 PRINT "{4RIGHT}***** ***** ***** *{5SPACE}*
     ****"
2070 PRINT "{4RIGHT}*{3SPACE}* *{3SPACE}* *{3SPA
     CE}* *{5SPACE}*
2080 PRINT "{4RIGHT}***** ***** ***** *{5SPACE}*
     **{2SPACE}"
2090 PRINT "{4RIGHT}*{3SPACE}* *{5SPACE}*{5SPACE
     }*{5SPACE}*{4SPACE}"
2100 PRINT "{4RIGHT}*{3SPACE}* *{5SPACE}*{5SPACE
     }***** *****"
2110 PRINT "{4RIGHT,2DOWN}***** ***** *{3SPACE}*
     {3SPACE}*{4SPACE}****"
2120 PRINT "{4RIGHT}*{3SPACE}* *{3SPACE}* **{2SP
     ACE}*{3SPACE}*{3SPACE}*{4SPACE}"
2130 PRINT "{4RIGHT}***** ***** **{2SPACE}*{3SPA
     CE}*{3SPACE}*"
2140 PRINT "{4RIGHT}*{5SPACE}*{3SPACE}* * * *{3S
     PACE}*{3SPACE}*"
2150 PRINT "{4RIGHT}*{5SPACE}*{3SPACE}* *{2SPACE
     }**{3SPACE}*{4SPACE}****"
2160 PRINT "{3DOWN,4RIGHT}<C> 1986
2170 PRINT "{4RIGHT,DOWN,2RIGHT}BY"
2180 PRINT "{3RIGHT,DOWN}ANDREAS ZILLA & A.K.{7S
     PACE}"
2190 PRINT "{3RIGHT,2DOWN,2RIGHT,SYNTH.:130}ZUM
     START BITTE 'S' DRUECKEN{SYNTH.:132}"
2200 GET T$: IF T$="" THEN 2200
2210 IF T$="S" THEN RETURN
2220 GOTO 2200
2230 END
2240 PRINT "{HOME,9DOWN,13RIGHT,SYNTH.:130}GAME
     OVER{SYNTH.:132}": FOR I=1 TO 2000: NEXT
2250 IF SC>HI THEN HI=SC
2260 GET A$: IF A$="" THEN 2260
2270 PRINT "{CLR}": RESTORE : GOTO 110
```

Listing 3. »Apple Panic« (Schluß)

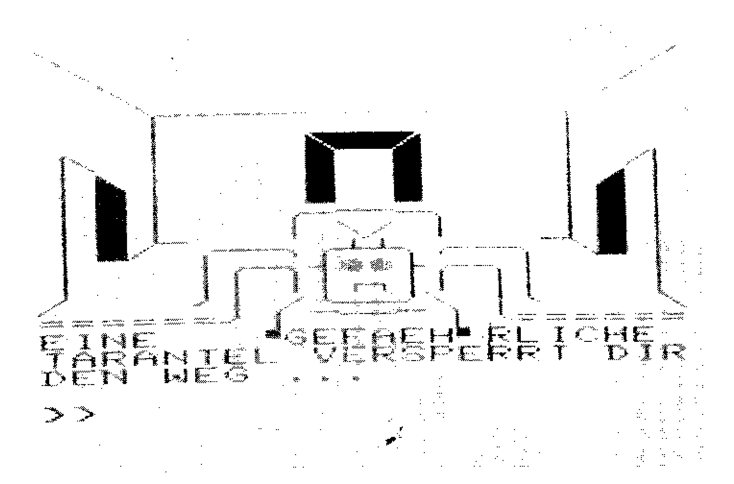
Gefährliche Pyramide!

Finden Sie die Maske des Pharao! Allein und eingeschlossen müssen Sie versuchen, der Pyramide ihr Geheimnis zu entreißen. Doch dabei sind viele Hindernisse zu überwinden. Sie benötigen nur einen VC 20 mit einer 16-KByte-Erweiterung.

Dangerous Pyramid (siehe Listing) ist ein Grafik-Adventure für den VC 20 mit mindestens 16 KByte Erweiterung. Sie sind in einer Pyramide eingeschlossen und müssen versuchen, sich der Maske des Pharao zu bemächtigen. Doch der Weg dahin ist weit und voller Gefahren.

Anweisungen an den Computer bestehen immer aus zwei Worten (zum Beispiel NIMM DIAMANT). Folgende Befehle stehen Ihnen zur Verfügung:

Besitz, Ende, Gehe, Nimm, Lege, Werfe, Oeffne, Zuende, Stecke, Trinke, Sage, Save (Spielstand), Springe, Load (Spielstand), Klettere und Schliesse.



Dahinter muß immer eine Richtung oder ein Gegenstand stehen. Genug der Anleitung, ab geht's in die Pyramide.

(S. Kaiser/og)

```
0 SW=1:PO=23:SA=0:SC=0:TR=0:TD=4:POKE 657,
  128:Q=36879:GOTO 1                           <115>
1 REM  ****************************           <013>
2 REM  *      DANGEROUS PYRAMID      *         <133>
3 REM  *      ================       *         <160>
4 REM  ****************************           <016>
5 REM  * EIN GRAFIK-ADVENTURE FUER *           <144>
6 REM  * DEN VC-20 UND:            *           <155>
7 REM  * - 16 KRAM ERWEITERUNG     *           <030>
8 REM  * - DATASETTE ODER FLOPPY   *           <055>
9 REM  ****************************           <021>
10 REM * PROGRAMMIERT  DEZ. 1984   *           <255>
11 REM *            BY             *           <115>
12 REM *        STEFAN KAISER      *           <132>
13 REM ****************************           <025>
14 :                                              <246>
15 :                                              <247>
16 REM <<< SPEICHERBEDARF: CA. 19,5  KBYTE
    >>>                                          <064>
17 :                                              <249>
18 :                                              <250>
19 PRINT CHR$(142):POKE Q,8:PRINT"{CLR,DOW
   N,RED}"TAB(12)"{RVSON}£{2SPACE}₸":PRINT
   "{2RIGHT,RVSON}£{2SPACE}₸{2RIGHT}£{RVOF
   F}£{2RIGHT,RVSON,SPACE}"                      <243>
20 PRINT"{2RIGHT,RVSON,SPACE,4RIGHT}£{RVOF
   F}£{3RIGHT}₸{RVSON,2SPACE}₸":PRINT"{2RI
   GHT}₸{RVSON,2SPACE}₸£ ₸{6RIGHT,SPACE,3R
   IGHT}£{RVOFF}£"                               <252>
21 PRINT"{3RIGHT,RVSON,SPACE,RIGHT,SPACE,R
   VOFF}£{RIGHT}₸{RVSON}₸{2RIGHT,RVOFF}₸{R
   VSON,2SPACE,RVOFF}£{2RIGHT,RVSON}£{RVOF
   F}£":PRINT"{2RIGHT}₸{RVSON,2SPACE,RVOFF
   }£{WHITE,3SPACE,RED}₸{RVSON}₸{2RIGHT}£₸
   {2RIGHT}£{RVOFF}£"                            <117>
22 PRINT"{3RIGHT,RVSON,SPACE,6RIGHT,RVOFF}
   ₸{RVSON}₸£{RVOFF}£₸{RVSON}₸£{RVOFF}£":P
   RINT"{3RIGHT,RVSON,SPACE,RVOFF,SPACE,WH
   ITE}1984{2SPACE,RED}₸£{2RIGHT}₸£"             <059>
23 PRINT TAB(6)"{PURPLE}@@@@@@@@@{DOWN,9LE
   FT,RVSON}PRESENTS:"                           <184>
24 PRINT"{2DOWN,GREEN,RIGHT,RVSON,SPACE}₸£
   ₸₸K£₸DID₸£₸{RVOFF}K{RVSON}K£₸":PRINT"{R
   IGHT}K{RVSON}KCV{2SPACE,RVOFF}KD{RVSON}
   D{RIGHT}C{RVOFF}£K{RVSON}K{RVOFF}K{RVSO
   N}KI{RVOFF}I"                                 <189>
25 PRINT"{RIGHT,RVSON,SPACE,RVOFF}£K{RVSON
   }K{RVOFF}K₸₸£{RVSON}C{RVOFF}IK{RVSON}₸{
   RVOFF}₸£₸£₸£"                                 <024>
26 PRINT"{YELLOW,SPACE}@@ @ @@@{2SPACE}@{5
   SPACE}@@@{2SPACE}N NVNVN NMNNVM{2SPACE}
   NN NM"                                        <023>
27 PRINT"{RVSON,2SPACE}₸{SPACE,RIGHT,SPACE
   }G ₸£ ₸₸{RVOFF,SPACE}M{RVSON}£GG ₸{RVOF
   F}NG{RVSON,SPACE,RVOFF}N{RVSON}H{SPACE,
   RVOFF}N{RVSON,SPACE}G{RVOFF}N{RVSON}H{S
   PACE,RVOFF}N{RVSON}H ₸£ GG{RIGHT,SPACE,
   RVOFF,SPACE}G";                               <194>
28 PRINT"{RVSON,2SPACE,RVOFF}£₸{RVSON,SPAC
   E,RVOFF}£{RVSON}G{SPACE,RVOFF}£{RVSON,2
   SPACE}H{4SPACE}GG{RIGHT,SPACE,RVOFF,SPA
   CE}G{RVSON,SPACE,RVOFF}H{2SPACE,RVSON,S
   PACE,RVOFF}H{RVSON}G{RVOFF}₸{RVSON}₸{SP
   ACE,RIGHT}H{4SPACE}GG{RVOFF}N{RVSON,SPA
   CE,RVOFF}N"                                   <221>
29 PRINT"{RVSON,SPACE,RVOFF}N{2SPACE,RVSON
   ,SPACE,RVOFF}N{RVSON}G{RVOFF}N{RVSON}H{
   SPACE,RVOFF}N{RVSON}H{SPACE,RVOFF}₸£{RV
   SON,SPACE}GG{SPACE,RVOFF}£{HOME}"             <007>
30 POKE Q,13:FOR A=1 TO 100:NEXT:GET A$:IF
    A$<>""THEN 32                                <222>
31 POKE Q,11:FOR A=1 TO 100:NEXT:GOTO 30        <238>
32 POKE 198,0:POKE Q,8                           <151>
33 DIM R$(51),G$(18),G(17),B$(16)                <036>
34 FOR A=1 TO 51:READ R$(A):NEXT:FOR A=1 T
   O 17:READ G$(A),G(A):NEXT                     <077>
35 FOR A=1 TO 4:READ F$(A),F(A):NEXT:FOR A
   =0 TO 2:FOR B=1 TO 4:READ W(A,B):NEXT B
   ,A                                            <119>
36 FOR A=1 TO 16:READ B$(A):NEXT:S$="{HOME
   ,16DOWN}":PRINT CHR$(142)                     <226>
37 GOSUB 263                                     <181>
38 Z=0:PRINT S$"{BLACK,RVSON}ICH SEHE:{DOW
   N}":FOR A=1 TO 17:IF G(A)=PO THEN 40          <251>
39 NEXT:GOTO 41                                  <108>
40 PRINT G$(A),:Z=1:GOTO 39                      <031>
41 IF Z=0 THEN PRINT"NICHTS"                     <255>
42 FOR A=1 TO 4:IF PO<>F(A)THEN NEXT:GOTO
   44                                            <048>
43 ON A GOSUB 339,333,326,337                    <254>
44 PRINT                                         <146>
45 A$="":POKE 19,64:INPUT"{BLUE}>>{20SPACE
   ,19LEFT,BLACK}";A$:POKE 19,0                  <031>
```

Listing »DANGEROUS PYRAMID« für den VC 20. Bitte beachten Sie die Eingabehinweise zum Abtippen von Listings auf Seite 129/130.

```
46 A1$=LEFT$(A$,4)                                    <006>
47 IF A1$=B$(11)THEN 61                               <055>
48 IF A1$=B$(12)THEN 67                               <067>
49 IF A1$=B$(15)THEN 219                              <145>
50 IF A1$=B$(16)THEN 242                              <021>
51 FOR A=1 TO LEN(A$):IF MID$(A$,A,1)<>" "
   THEN NEXT:GOTO 59                                  <167>
52 A1$=LEFT$(A$,A-1):A2$=RIGHT$(A$,LEN(A$)
   -A):A3$=LEFT$(A1$,4):A4$=LEFT$(A2$,4)              <148>
53 FOR A=1 TO 16:IF B$(A)=A3$THEN 55                  <250>
54 NEXT:GOTO 56                                       <220>
55 ON A GOTO 69,114,87,121,143,167,186,200
   ,218,225,61,67,253,248,219,242                     <193>
56 GOSUB 414:PRINT S$"{DOWN,RED}WAS IST DA
   S :"                                               <208>
57 PRINT"{RVSON}"A1$"N{RVOFF,SPACE,RED}?"             <098>
58 PRINT"DIESER BEFEHL IST MIR NICHT BEKAN
   NT !!!":GOSUB 410:GOTO 37                          <192>
59 GOSUB 414:PRINT S$"{2DOWN,RED}EIN BEFEH
   L BESTEHT{4SPACE}IMMER AUS{SPACE,RVSON}
   2{RVOFF,SPACE}WOERTERN"                            <055>
60 GOSUB 410:GOTO 37                                  <031>
61 PRINT"{CLR,BLUE,RVSON,SPACE}DAS IST DEI
   N BESITZ: "                                        <082>
62 Z=0:PRINT:FOR A=1 TO 17:IF G(A)=52 THEN
   PRINT"{BLACK}"G$(A):Z=1                            <117>
63 NEXT:IF Z=0 THEN PRINT"{BLACK}NICHTS"              <074>
64 PRINT"{2DOWN,BLUE,RVSON,SPACE}DAS IST D
   EIN BESITZ: "                                      <191>
65 IF Y=1 THEN RETURN                                 <186>
66 GOSUB 410:GOTO 37                                  <037>
67 PRINT"{CLR}":GOSUB 305:POKE Q-12,40                <195>
68 PRINT S$"{2DOWN,BLACK}@@@@@@@@@@@@@@@@@
   @@@@@{RVSON,SPACE}AUF WIEDERSEHEN ....{
   SPACE,YELLOW}";:END                                <026>
69 FOR A=1 TO 4:IF MID$(R$(PO),A,1)<>LEFT$
   (A4$,1)THEN NEXT:GOTO 77                           <219>
70 IF PO=36 AND LEFT$(A4$,1)="S"THEN W=4:G
   OTO 399                                            <144>
71 IF PO=46 AND G(5)<>0 AND LEFT$(A4$,1)="
   S"THEN W=4:GOTO 399                                <228>
72 IF PO=39 AND F(1)=39 THEN W=1:GOTO 388             <250>
73 IF PO=50 AND F(2)=50 THEN W=2:GOTO 388             <005>
74 IF PO=43 AND F(3)=43 THEN W=3:GOTO 388             <152>
75 IF PO=1 AND LEFT$(A4$,1)<>"R"THEN W=4:G
   OTO 399                                            <098>
76 PO=PO+W(SW,A):GOTO 37                              <046>
77 IF LEFT$(A4$,1)=MID$(R$(PO),6,1)THEN IF
   PO=1 OR PO=10 OR PO=11 THEN 80                     <234>
78 IF LEFT$(A4$,1)=MID$(R$(PO),5,1)THEN 84            <251>
79 GOTO 86                                            <153>
80 SW=SW-1:IF PO=1 THEN PO=8                          <205>
81 IF PO=10 THEN PO=21                                <017>
82 IF PO=11 THEN PO=27                                <082>
83 GOTO 37                                            <133>
84 GOSUB 414:PRINT S$"{2DOWN,RED}DU KANNST
   DOCH NICHT{2SPACE}DIE WAENDE HOCHGEHEN
   !"                                                 <248>
85 GOSUB 410:GOTO 37                                  <056>
86 GOSUB 414:PRINT S$"{2DOWN,RED,SPACE}DIE
   SE RICHTUNG GIBT{3SPACE}ES HIER NICHT !
   !!":GOSUB 410:GOTO 37                              <229>
87 FOR A=1 TO 17:IF A4$<>LEFT$(G$(A),4)THE
   N NEXT:GOTO 89                                     <253>
88 GOTO 93                                            <122>
89 FOR B=1 TO 4:IF A4$<>LEFT$(F$(B),4)THEN
   NEXT:GOTO 96                                       <022>
90 IF PO=33 AND B=4 THEN 110                          <091>
91 IF PO=F(B)THEN W=B:GOTO 388                        <046>
92 GOTO 96                                            <174>
93 IF A<4 THEN 106                                    <115>
94 IF A<6 THEN 109                                    <186>
95 IF G(A)=PO THEN 98                                 <236>
96 GOSUB 414:PRINT S$"{DOWN,RED}ICH GLAUBE
   , DU HAST{3SPACE}TOMATEN AUF DEN AUGEN.
   ";                                                 <213>
97 PRINT"DIESEN GEGENSTAND GIBTES HIER DOC
   H GAR NICHT";:GOSUB 410:GOTO 37                    <238>
98 IF TR>=TD+SA THEN SK=TD+SA:GOSUB 414:GO
   TO 105                                             <141>
99 IF A=9 AND G(7)=52 THEN W=5:Z=0:GOTO 38
   4                                                  <067>
100 IF A=7 AND G(9)=52 THEN W=5:Z=0:GOTO 3
    84                                                <007>
101 IF A=10 THEN SA=2                                 <253>
102 IF A=15 AND PH<>1 THEN 112                        <107>
103 IF A=15 AND PH=1 THEN 369                         <080>
104 TR=TR+1:G(A)=52:GOTO 61                           <251>
105 PRINT S$"{2DOWN,RED}DU DARFST DOCH NUR
    {BLUE,3SPACE}"SK"{RED}GEGENSTAENDE TRA
    GEN.":GOSUB 410:GOTO 37                           <095>
106 IF A=1 AND PO<>G(1)THEN A=3                       <243>
107 IF PO<>G(A)THEN 96                                <120>
108 W=6:GOTO 391                                      <158>
109 IF PO<>G(A)AND PO<>F(B)THEN 96                    <041>
110 GOSUB 414:PRINT S$"{2DOWN,RED}ABER DAS
    KANN MAN DOCHNICHT NEHMEN !!!"                    <010>
111 GOSUB 410:GOTO 37                                 <082>
112 GOSUB 414:PRINT S$:PRINT"{2DOWN,RED}DI
    E "G$(15)" KANNST DU{3SPACE}JETZT NOCH
    NICHT"                                            <044>
113 PRINT"NEHMEN !!!":GOSUB 410:GOTO 37               <054>
114 FOR A=6 TO 17:IF LEFT$(G$(A),4)<>A4$TH
    EN NEXT:GOTO 119                                  <241>
115 IF A=13 AND G(A)=52 AND PO=46 THEN G(A
    )=0:G(5)=0:TR=TR-1:GOTO 37                        <231>
116 IF A=10 AND G(A)=52 THEN SA=0                     <021>
117 IF G(A)=52 THEN G(A)=PO:TR=TR-1:GOTO 3
    7                                                 <115>
118 GOSUB 414:PRINT S$"{2DOWN,RED}DIESEN G
    EGENSTAND HASTDU ABER GAR NICHT !!":GO
    SUB 410:GOTO 37                                   <058>
119 GOSUB 414:PRINT S$"{DOWN,RED}WAS IST D
    ENN DAS FUER EIN ZEUG, DAS DU DA"                 <022>
120 PRINT"ABLEGEN WILLST ???":GOSUB 410:GO
    TO 37                                             <040>
121 FOR A=6 TO 17:IF A4$<>LEFT$(G$(A),4)TH
    EN NEXT:GOTO 125                                  <175>
122 IF PO=39 OR PO=50 OR PO=43 THEN 127               <143>
123 IF G(A)=52 THEN G(A)=PO:TR=TR-1:GOTO 3
    7                                                 <121>
124 GOTO 118                                          <069>
125 GOSUB 414:PRINT S$"{DOWN,RED}WAS IST D
    ENN DAS FUER EIN ZEUG, DAS DU DA"                 <028>
126 PRINT"WEGWERFEN WILLST ???":GOSUB 410:
    GOTO 37                                           <048>
127 IF PO=39 AND F(1)=39 AND G(17)<>52 THE
    N W=1:GOTO 388                                    <110>
128 IF PO=39 AND F(1)=39 AND A4$<>LEFT$(G$
    (17),4)THEN W=1:GOTO 388                          <075>
129 IF PO=50 AND F(2)=50 AND G(8)<>52 THEN
    W=2:GOTO 388                                      <197>
130 IF PO=50 AND F(2)=50 AND A4$<>LEFT$(G$
    (8),4)THEN W=2:GOTO 388                           <191>
131 IF PO=43 AND F(3)=43 AND G(16)<>52 THE
    N W=3:GOTO 388                                    <046>
132 IF PO=43 AND F(3)=43 AND A4$<>LEFT$(G$
    (16),4)THEN W=3:GOTO 388                          <235>
133 IF PO=39 THEN G(17)=0:F(1)=0:TR=TR-1:G
    OTO 136                                           <112>
134 IF PO=50 THEN G(8)=0:F(2)=0:TR=TR-1:GO
    TO 138                                            <095>
135 IF PO=43 THEN G(16)=0:F(3)=0:TR=TR-1:G
    OTO 140                                           <132>
136 GOSUB 414:PRINT S$"{PURPLE}DAS GLEISSE
    NDE LICHT{2SPACE}DES "G$(17)"EN HAT DI
    E"                                                <033>
137 PRINT"GEFAEHRLICHE MUMIE ZU DEINEM GLU
    ECK VER-{4SPACE}NICHTET !!!";:GOSUB 41
    0:GOTO 37                                         <116>
138 GOSUB 414:PRINT S$"{DOWN,PURPLE}DER "G
    $(8)" STECKT JETZTIM KOPF DER "F$(2)"
    ,";                                               <235>
139 PRINT"D.H. SIE IST NUN TOT !";:GOSUB 4
    10:GOTO 37                                        <255>
140 GOSUB 414:PRINT S$"{DOWN,PURPLE}DU HAS
    T DAS "G$(16)" GENAU";                            <233>
141 PRINT"IN DEN RACHEN DER{5SPACE}"F$(3)"
    GEWORFEN !"                                       <088>
142 PRINT"SIE IST ELEND ZUGRUNDEGEGANGEN .
    ..";:GOSUB 410:GOTO 37                            <037>
143 FOR A=1 TO 17:IF LEFT$(G$(A),4)<>A4$TH
    EN NEXT:GOTO 199                                  <114>
144 IF A>3 AND A<>10 THEN 158                         <056>
145 IF PO<>4 THEN 152                                 <159>
146 IF A=1 AND G(3)=4 THEN 161                        <063>
147 IF A=1 AND SC<>1 THEN 163                         <111>
148 IF A=1 AND SC=1 THEN G(1)=0:G(3)=4                <166>
```

```
149 IF A=1 AND G(6)=0 THEN G(6)=4                      <072>
150 IF A<>1 THEN 160                                   <187>
151 GOTO 37                                            <201>
152 IF PO=G(10)OR G(10)=52 THEN IF A=10 TH
    EN 165                                             <086>
153 IF PO<>39 THEN 157                                 <032>
154 IF F(1)=0 AND A=2 THEN F(1)=39:GOTO 37             <028>
155 IF F(1)=39 THEN W=1:GOTO 388                       <105>
156 IF F(1)=52 AND A=2 THEN 159                        <065>
157 GOTO 160                                           <181>
158 GOSUB 414:PRINT S$"{2DOWN,RED}DAS KANN
    ST DU NICHT{3SPACE}OEFFNEN !!!":GOSUB
    410:GOTO 37                                        <126>
159 GOSUB 414:PRINT S$"{2DOWN,RED}DER "G$(
    2)" ISTSCHON OFFEN !!!":GOSUB 410:GOTO
    37                                                 <218>
160 GOSUB 414:PRINT S$"{2DOWN,RED}SO ETWAS
    GIBT ES HIER LEIDER NICHT !";:GOSUB 4
    10:GOTO 37                                         <056>
161 GOSUB 414:PRINT S$"{2DOWN,RED}DIE "G$(
    1)" IST{3SPACE}DOCH SCHON OFFEN, DU{2S
    PACE}BLINDSCHLEICHE !";                            <076>
162 GOSUB 410:GOTO 37                                  <133>
163 GOSUB 414:PRINT S$"{2DOWN,RED}DU MUSST
    ERST EINMAL{2SPACE}DIE TRUHE AUF-"                 <192>
164 PRINT"SCHLIESSEN !!!";:GOSUB 410:GOTO
    37                                                 <085>
165 GOSUB 414:PRINT S$"{2DOWN,PURPLE}JUHU,
    DU SIEHST ETWAS HERRLICHES, NAEMLICH:
    "                                                  <181>
166 FOR A=1 TO 5000:NEXT:PRINT"{RVSON}NICH
    TS{RVOFF,SPACE}...";:GOSUB 410:GOTO 37             <145>
167 IF G(12)<>52 THEN 182                              <025>
168 IF A4$=LEFT$(G$(7),4)THEN 172                      <019>
169 IF A4$=LEFT$(G$(9),4)THEN 176                      <116>
170 IF A4$=LEFT$(G$(12),4)THEN 180                     <047>
171 GOSUB 414:PRINT S$"{2DOWN,RED}SO ETWAS
    KANNST DU{4SPACE}NICHT ANZUENDEN !":G
    OSUB 410:GOTO 37                                   <018>
172 IF G(7)=52 THEN W=5:GOTO 384                       <166>
173 IF G(7)=PO THEN Z=1:G(7)=0:GOSUB 384:G
    OTO 37                                             <002>
174 IF G(7)=0 AND PO=33 AND DY=1 THEN Z=1:
    GOSUB 384:R$(33)="N.OW..":F(4)=0:GOTO
    37                                                 <255>
175 GOTO 184                                           <103>
176 IF PO=G(9)OR G(9)=52 THEN 178                      <156>
177 GOTO 184                                           <105>
178 GOSUB 414:PRINT S$"{DOWN,PURPLE}DIE "G
    $(9)" BRENNT DOCHSCHON, DU BRAUCHST SI
    E";                                                <179>
179 PRINT"ALSO NICHT MEHR ANZU- ZUENDEN.";
    :GOSUB 410:GOTO 37                                 <001>
180 GOSUB 414:PRINT S$"{DOWN,PURPLE}DAS HA
    ST DU ABER FEIN GEMACHT ! DAS STREICH-
    ";                                                 <105>
181 PRINT"HOLZ BRENNT NUN.":GOSUB 410:GOTO
    37                                                 <136>
182 GOSUB 414:PRINT S$"{DOWN,RED}DU HAST D
    OCH GAR{6SPACE}NICHTS, UM DIESEN"                  <044>
183 PRINT"GEGENSTAND ANZUZUENDEN";:GOSUB 4
    10:GOTO 37                                         <185>
184 GOSUB 414:PRINT S$"{DOWN,RED}WO SOLL D
    ENN DIESER{3SPACE}GEGENSTAND HIER"                 <186>
185 PRINT"LIEGEN ???";:GOSUB 410:GOTO 37               <126>
186 IF PO<>4 AND PO<>33 AND PO<>38 THEN 19
    4                                                  <076>
187 IF PO=4 AND A4$=LEFT$(G$(11),4)AND G(1
    1)=52 THEN SC=1:TR=TR-1:G(11)=0:GOTO 3
    7                                                  <027>
188 IF PO=33 AND A4$=LEFT$(G$(7),4)AND G(7
    )=52 THEN DY=1:TR=TR-1:G(7)=0:GOTO 37              <200>
189 IF PO=38 AND A4$=LEFT$(G$(6),4)AND G(6
    )=52 THEN PH=1:TR=TR-1:G(6)=0:GOTO 37              <174>
190 IF PO=4 AND A4$=LEFT$(G$(11),4)AND G(1
    1)<>52 THEN 118                                    <033>
191 IF PO=33 AND A4$=LEFT$(G$(7),4)AND G(7
    )<>52 THEN 118                                     <084>
192 IF PO=38 AND A4$=LEFT$(G$(6),4)AND G(6
    )<>52 THEN 118                                     <208>
193 GOTO 196                                           <201>
194 GOSUB 414:PRINT S$"{DOWN,RED}ES GIBT H
    IER NICHTS,{2SPACE}WOHIN DU DEN GEGEN-
    "                                                  <233>
195 PRINT"STAND HINSTECKEN{6SPACE}KOENNTES
    T !":GOSUB 410:GOTO 37                             <179>
196 FOR A=6 TO 17:IF A4$<>LEFT$(G$(A),4)TH
    EN NEXT:GOTO 199                                   <129>
197 IF G(A)<>52 THEN 118                               <097>
198 GOSUB 414:PRINT S$"{3DOWN,RED}DAS GEHT
    LEIDER NICHT.";:GOSUB 410:GOTO 37                  <225>
199 GOSUB 414:PRINT S$"{3DOWN,RED}WAS IST
    DENN DAS ???":GOSUB 410:GOTO 37                    <101>
200 IF A4$<>LEFT$(G$(14),4)AND A4$<>LEFT$(
    G$(16),4)THEN 204                                  <165>
201 IF A4$=LEFT$(G$(14),4)AND G(14)=52 THE
    N 205                                              <182>
202 IF A4$=LEFT$(G$(16),4)AND G(16)=52 THE
    N W=7:GOTO 391                                     <018>
203 GOTO 118                                           <148>
204 GOSUB 414:PRINT S$"{2DOWN,RED}IGITT !
    HAST DU EINEN GESCHMACK !!!":GOSUB 410
    :GOTO 37                                           <064>
205 PRINT"{CLR,RED}DER "G$(14)" VER-":PRIN
    T"{DOWN}ZAUBERT DICH IN EINEN{SPACE,DO
    WN}ZWERG. DIE KONSEQUENZ"                          <160>
206 PRINT"{DOWN}DARAUS IST, DASS DU{3SPACE
    ,DOWN}NUR NOCH HALB SOVIEL{2SPACE,DOWN
    }TRAGEN KANNST, WIE"                               <038>
207 PRINT"{DOWN}VORHER !!!":G(14)=0:TR=TR-
    1                                                  <115>
208 IF(TR-SA)>(TD/2)THEN 210                           <024>
209 TD=TD/2:GOSUB 410:GOTO 37                          <105>
210 GOSUB 410:Y=1:GOSUB 61:Y=0                         <057>
211 PRINT"{DOWN,RED}DU MUSST VON DEINEM{3S
    PACE}BESITZ{SPACE,PURPLE}1{RED,SPACE}G
    EGEN-"                                             <037>
212 PRINT"STAND ABGEBEN !!{6SPACE,DOWN,GRE
    EN}WELCHEN ???{DOWN}"                              <057>
213 PRINT:POKE 19,64:INPUT"{BLUE}";C$:POKE
    19,0                                               <119>
214 FOR B=6 TO 17:IF LEFT$(C$,4)<>LEFT$(G$
    (B),4)THEN NEXT:GOTO 217                           <113>
215 IF G(B)=52 THEN TR=TR-1:G(B)=0:GOTO 20
    9                                                  <095>
216 PRINT"{CLR,RED}KANN ES VIELLEICHT{4SPA
    CE}SEIN, DASS DU DAS GAR NICHT BESITZT
    ??":GOTO 213                                       <195>
217 PRINT"{CLR,RED}WAS IST DENN DAS ???{2S
    PACE}GIB BITTE ETWAS AN,{3SPACE}DAS DU
    AUCH BESITZT !":GOTO 213                           <202>
218 GOSUB 414:PRINT S$"{DOWN,GREEN}''A2$''
    ":GOSUB 410:GOTO 37                                <059>
219 POKE Q,24:PRINT"{CLR,RVSON,BLACK}SPIEL
    STAND ABSPEICHERN{RVOFF,BLUE}"                     <138>
220 PRINT"{2DOWN,SPACE}PRESS{SPACE,RVSON}R
    ECORD{RVOFF,SPACE}&{SPACE,RVSON}PLAY{R
    VOFF,9SPACE,DOWN}ON TAPE"                          <173>
221 WAIT 37151,64,64:PRINT"{3UP,PURPLE,5SP
    ACE}BITTE WARTEN{5SPACE,DOWN,14SPACE,2
    DOWN}"                                             <226>
222 OPEN 1,1,1,"DATEN":FOR A=1 TO 17:PRINT
    #1,G(A):NEXT:FOR A=1 TO 4:PRINT#1,F(A)
    :NEXT                                              <135>
223 PRINT#1,PO:PRINT#1,SW:PRINT#1,TR:PRINT
    #1,TD:PRINT#1,SA:PRINT#1,SC:PRINT#1,PH             <188>
224 PRINT#1,DY:CLOSE 1:PRINT"{RED,SPACE}=>
    TASTE DRUECKEN <=":POKE 198,0:WAIT 19
    8,1:GOTO 37                                        <184>
225 FOR A=1 TO 4:IF MID$(R$(PO),A,1)<>LEFT
    $(A4$,1)THEN NEXT:GOTO 232                         <239>
226 IF PO=46 AND G(13)<>PO AND LEFT$(A4$,1
    )="S"THEN W=4:GOTO 399                             <052>
227 IF PO=39 AND F(1)=PO THEN W=1:GOTO 388             <232>
228 IF PO=50 AND F(2)=PO THEN W=2:GOTO 388             <246>
229 IF PO=43 AND F(3)=PO THEN W=3:GOTO 388             <072>
230 IF PO=1 AND LEFT$(A4$,1)<>"R"THEN W=4:
    GOTO 399                                           <253>
231 PO=PO+W(SW,A):GOTO 37                              <201>
232 IF LEFT$(A4$,1)=MID$(R$(PO),6,1)THEN I
    F PO=1 OR PO=10 OR PO=11 THEN 80                   <133>
233 IF LEFT$(A4$,1)=MID$(R$(PO),5,1)THEN 2
    38                                                 <249>
```

Listing »DANGEROUS PYRAMID« (Fortsetzung)

```
234 IF LEFT$(A4$,1)="H"AND MID$(R$(PO),5,1
    )<>"H"THEN 236                          <247>
235 GOTO 86                                 <053>
236 POKE Q,154:POKE Q-1,15:POKE Q-2,200:FO
    R A=1 TO 1500:NEXT:FOR A=15 TO 0 STEP-
    .1                                      <213>
237 POKE Q-1,A:NEXT:POKE Q-2,0:W=9:GOTO 39
    1                                       <117>
238 SW=SW+1:IF PO=8 THEN PO=1               <117>
239 IF PO=21 THEN PO=10                     <063>
240 IF PO=27 THEN PO=11                     <099>
241 GOTO 37                                 <035>
242 POKE Q,30:PRINT"{CLR,RVSON,BLUE,SPACE}
    SPIELSTAND{2SPACE}EINLADEN{SPACE,RVOFF
    ,BLACK}"                                <017>
243 PRINT"{2DOWN,2SPACE}PRESS{SPACE,RVSON}
    PLAY{RVOFF,SPACE}ON TAPE"               <183>
244 WAIT 37151,64,64:PRINT"{UP,PURPLE,5SPA
    CE}BITTE WARTEN{4SPACE,WHITE,2DOWN}"    <254>
245 OPEN 1,1,0,"DATEN":FOR A=1 TO 17:INPUT
    #1,G(A):NEXT:FOR A=1 TO 4:INPUT#1,F(A)
    :NEXT                                   <166>
246 INPUT#1,PO:INPUT#1,SW:INPUT#1,TR:INPUT
    #1,TD:INPUT#1,SA:INPUT#1,SC:INPUT#1,PH  <192>
247 INPUT#1,DY:CLOSE 1:PRINT"{RED,SPACE}=>
     TASTE DRUECKEN <=":POKE 198,0:WAIT 19
    8,1:GOTO 37                             <127>
248 FOR A=1 TO 6:IF MID$(R$(PO),A,1)<>LEFT
    $(A4$,1)THEN NEXT:GOTO 86               <112>
249 IF A<>5 THEN 251                        <034>
250 GOTO 238                                <235>
251 GOSUB 414:PRINT S$"{DOWN,RED}IN DIESE
    RICHTUNG{5SPACE}BRAUCHST DU NICHT"      <144>
252 PRINT"ZU KLETTERN, DENN DU{2SPACE}KANN
    ST DAHIN GEHEN !!";:GOSUB 410:GOTO 37   <186>
253 FOR A=1 TO 3:IF A4$<>LEFT$(G$(A),4)THE
    N NEXT:GOTO 260                         <090>
254 IF A=1 AND PO=4 AND G(1)=4 THEN 261     <198>
255 IF A=2 AND PO=39 AND F(1)=0 THEN 262    <104>
256 IF A=2 AND PO=39 AND F(1)=PO THEN W=1:
    GOTO 388                                <185>
257 IF A=1 THEN A=3                         <045>
258 IF A=3 AND PO=4 THEN G(1)=4:G(3)=0:G(1
    1)=4:SC=0:GOTO 37                       <210>
259 GOTO 160                                <029>
260 GOSUB 414:PRINT S$"{2DOWN,RED}DAS KANN
    ST DU ABER GARNICHT SCHLIESSEN !!!":GO
    SUB 410:GOTO 37                         <228>
261 GOSUB 414:PRINT S$"{2DOWN,RED}DIE "G$(
    1)" IST{3SPACE}SCHON ZU !!!":GOSUB 410
    :GOTO 37                                <205>
262 GOSUB 414:PRINT S$"{2DOWN,RED}DER "G$(
    2)" IST{5SPACE}SCHON ZU !!!":GOSUB 410
    :GOTO 37                                <249>
263 IF PO=1 THEN 344                        <141>
264 POKE Q,25                               <071>
265 PRINT"{CLR,BLACK}M{20SPACE}N M{18SPACE
    }N{3SPACE}M{16SPACE}N"                  <224>
266 PRINT"{3SPACE}MGGGGGGGGGGGGGGGGN"       <130>
267 FOR A=1 TO 7:PRINT"{3SPACE}H{14SPACE}G
    ":NEXT                                  <209>
268 PRINT"{3SPACE}NTTTTTTTTTTTTTTTM"        <162>
269 PRINT"{2SPACE}N{16SPACE}M{3SPACE}N{18S
    PACE}M N{20SPACE}M";                    <142>
270 FOR A=1 TO 22:PRINT"{RED}=";:NEXT       <156>
271 IF MID$(R$(PO),1,1)<>"N"THEN 275        <107>
272 PRINT"{HOME,5DOWN}"TAB(9)"{BLACK,RVSON
    }M{2SPACE}N{DOWN,4LEFT,SPACE,RVOFF,2SP
    ACE}";                                  <251>
273 PRINT"{HOME,5DOWN}"TAB(9)"{RVSON,SPACE
    ,DOWN,4LEFT,SPACE,RVOFF,2SPACE,RVSON,S
    PACE,DOWN,4LEFT,SPACE,RVOFF,2SPACE,RVS
    ON,SPACE,DOWN,4LEFT,SPACE,RVOFF,2SPACE
    ,RVSON,SPACE,DOWN,4LEFT}";              <249>
274 PRINT"{RVOFF}£{2SPACE}*{DOWN,4LEFT,4SP
    ACE,HOME}"                              <023>
275 IF MID$(R$(PO),3,1)<>"O"THEN 278        <154>
276 PRINT"{HOME,6DOWN}"TAB(20)"{BLACK}NG{D
    OWN,3LEFT,RVSON}£{RVOFF,SPACE}G{DOWN,3
    LEFT,RVSON,SPACE,RVOFF,SPACE}G{DOWN,3L
    EFT,RVSON,SPACE,RVOFF,SPACE}G{DOWN,3LE
    FT,RVSON,SPACE,RVOFF,SPACE}G{DOWN,3LEF
    T,RVSON,SPACE,RVOFF,SPACE}G{DOWN,3LEFT
    ,2SPACE}G";                             <088>
277 PRINT"{DOWN,2LEFT,SPACE}G{HOME}"        <226>
278 IF MID$(R$(PO),4,1)<>"W"THEN 280        <102>
279 PRINT"{HOME,6DOWN,BLACK}HM{DOWN,2LEFT}
    H{SPACE,RVSON}*{RVOFF,DOWN,3LEFT}H{SPA
    CE,RVSON,SPACE,RVOFF,DOWN,3LEFT}H{SPAC
    E,RVSON,SPACE,RVOFF,DOWN,3LEFT}H{SPACE
    ,RVSON,SPACE,RVOFF,DOWN,3LEFT}H{SPACE,
    RVSON,SPACE,RVOFF,DOWN,3LEFT}H{2SPACE,
    DOWN,3LEFT}H{SPACE,HOME}"               <207>
280 IF MID$(R$(PO),5,1)<>"H"THEN 282        <002>
281 PRINT"{HOME}"TAB(7)"{BLACK}GGGGGGGG{DO
    WN,8LEFT}*{RVSON}*{RVOFF,4SPACE,RVSON}
    £{RVOFF}£{DOWN,7LEFT}*{RVSON}G{2SPACE}
    H{RVOFF}£{HOME}"                        <036>
282 IF MID$(R$(PO),6,1)<>"R"THEN 284        <102>
283 PRINT"{HOME}"TAB(8)"{12DOWN,BLACK,RVSO
    N}£G{2SPACE}H*{DOWN,7LEFT}£{RVOFF}£{4S
    PACE}*{RVSON}*{RVOFF,DOWN,8LEFT}TTTTTT
    TT{HOME}"                               <187>
284 FOR A=1 TO 17:IF PO<>G(A)THEN 288       <215>
285 ON A GOSUB 310,317,313,322,324,320,289
    ,290,291,293                            <099>
286 IF A<10 THEN 288                        <026>
287 ON A-10 GOSUB 296,297,299,301,305,302,
    303                                     <179>
288 NEXT:RETURN                             <098>
289 PRINT S$TAB(15)"{3UP,RVSON,RED,2SPACE,
    RVOFF,BLACK}I{HOME}":RETURN             <157>
290 PRINT S$TAB(5)"{3UP,RVSON,PURPLE}**-{S
    PACE,UP,LEFT}*{2DOWN,LEFT,RVOFF}£{RVSO
    N,UP,YELLOW,SPACE}TYU{RVOFF}TOPG{HOME}
    ":RETURN                                <200>
291 PRINT S$TAB(14)"{2UP,BLACK}HH{UP,2LEFT
    }TT{UP,2LEFT,RVSON}*{RVOFF}*{UP,2LEFT,
    RVSON}*{RVOFF}*{UP,2LEFT,RVSON}*{RVOFF
    }*{UP,2LEFT,YELLOW}*£{UP,2LEFT,RVSON,2
    SPACE,UP,3LEFT,RVOFF}*{RVSON,2SPACE,RV
    OFF}£";                                 <064>
292 PRINT"{UP,4LEFT,RVSON,4SPACE,UP,4LEFT}
    £{2SPACE}*{UP,3LEFT}*£{HOME}":RETURN    <083>
293 PRINT"{HOME}"TAB(12)"{2DOWN,BLACK}OPOP
    G{DOWN,5LEFT}*{RVSON,3SPACE,RVOFF}£{DO
    WN,4LEFT}*{RVSON,SPACE,RVOFF}£{DOWN,3L
    EFT,RVSON}£Y*{DOWN,4LEFT}£{3SPACE}*{DO
    WN,6LEFT,RVOFF}H{RVSON,3SPACE}";        <076>
294 PRINT"{2SPACE,RVOFF}G{DOWN,7LEFT}H{RVS
    ON,5SPACE,RVOFF}H{DOWN,7LEFT}T{RVSON,5
    SPACE,RVOFF}T{DOWN,7LEFT}H{RVSON,5SPAC
    E,RVOFF}H{DOWN,7LEFT}H{RVSON,5SPACE,RV
    OFF}G";                                 <147>
295 PRINT"{DOWN,6LEFT}*{RVSON,3SPACE,RVOFF
    }£{DOWN,4LEFT}TYT{HOME}":RETURN         <245>
296 PRINT S$"{6UP,3RIGHT,BLACK}TTT{YELLOW}
    H{BLACK}B{YELLOW}-{UP,3LEFT}U*I{2DOWN,
    3LEFT}J*K{6LEFT,BLACK,RVSON}G{HOME}":R
    ETURN                                   <069>
297 PRINT S$TAB(12)"{4UP,PURPLE}GGGGG{DOWN
    ,6LEFT,RVOFF}N{4SPACE,RVSON}£{RVOFF,DO
    WN,6LEFT}";                             <150>
298 PRINT"{YELLOW}*****{PURPLE}£{HOME}":RE
    TURN                                    <088>
299 PRINT S$TAB(8)"{6UP,BLACK}GGGGGGGG{DOW
    N,9LEFT}N{7SPACE}NG{DOWN,11LEFT}N{7SPA
    CE}NN";                                 <168>
300 PRINT"{DOWN,10LEFT}OTTTTTTPN{DOWN,9LEF
    T}TTTTTTTT{HOME}":RETURN                <164>
301 PRINT S$TAB(4)"{7UP,GREEN}HS{DOWN,3LEF
    T}UKJI{DOWN,4LEFT}-{RED}DF{GREEN}-{DOW
    N,4LEFT}-{RED}CV{GREEN}-{DOWN,4LEFT}Z*
    *X{HOME}":RETURN                        <234>
302 PRINT S$TAB(4)"{7UP,GREEN}HS{DOWN,3LEF
    T}UKJI{DOWN,4LEFT}-{BLACK}DF{GREEN}-{D
    OWN,4LEFT}-{BLACK}CV{GREEN}-{DOWN,4LEF
    T}Z**X{HOME}":RETURN                    <028>
303 PRINT S$TAB(12)"{8UP,YELLOW}G{DOWN,2LE
    FT}N M{DOWN,4LEFT}NMGNM{DOWN,6LEFT}NMN
    MNM{DOWN,7LEFT}G M N H{DOWN,7LEFT}";    <192>
304 PRINT"MGNTMGN{HOME}":RETURN             <146>
305 POKE Q,120:PRINT"{HOME}"TAB(11)"{RVSON
    ,GREEN}*{DOWN,3LEFT,RVOFF,BLACK}NT{RVS
    ON,GREEN,SPACE,RVOFF,BLACK}TM{DOWN,6LE
    FT,BLUE,RVSON}£{5SPACE}*{DOWN,8LEFT,BL
    ACK,RVOFF}N N{3SPACE}M M";              <138>
306 PRINT"{DOWN,9LEFT,RVSON,BLUE,2SPACE,RV
```

```
    OFF,BLACK,SPACE}B B{SPACE,RVSON,BLUE,2
    SPACE,RVOFF,DOWN,10LEFT}N R{2SPACE,BLU
    E}-{2SPACE,BLACK}G M{DOWN,11LEFT,RVSON
    ,BLUE}";                                  <156>
307 PRINT"{3SPACE,RVOFF,2SPACE}F{2SPACE,RV
    SON,BLUE,3SPACE,RVOFF,DOWN,11LEFT,BLAC
    K}G R{5SPACE}G R{DOWN,11LEFT,RVSON,BLU
    E,3SPACE,RVOFF,SPACE,RED}TTF{SPACE,RVS
    ON,BLUE,3SPACE,RVOFF}";                   <074>
308 PRINT"{DOWN,11LEFT,BLACK}G{2SPACE}MGGG
    N{2SPACE}R{DOWN,11LEFT,BLUE}*{RVSON,2S
    PACE,RVOFF}£{SPACE,BLACK,RVSON}T{RVOFF
    ,SPACE,BLUE}*{RVSON,2SPACE,RVOFF}£{DOW
    N,10LEFT}";                               <212>
309 PRINT"{BLACK}M{3SPACE,BLACK,RVSON}T{RV
    OFF,3SPACE}N{DOWN,8LEFT}M{SPACE,CYAN,R
    VSON}TB{RVOFF,2SPACE,BLACK}N{DOWN,6LEF
    T}MGGGN{HOME}":RETURN                     <200>
310 PRINT"{HOME}"TAB(8)"{7DOWN,BLACK,RVSON
    }£{5SPACE}N{RVOFF}G{DOWN,9LEFT,RVSON}£
    {5SPACE}N{RVOFF}£G{DOWN,9LEFT,RVSON}TT
    TTT£{RVOFF}£ G";                          <033>
311 PRINT"{DOWN,9LEFT}G{SPACE,PURPLE}R{2SP
    ACE,BLACK}R{2SPACE}G{DOWN,9LEFT}G{PURP
    LE}G{GREEN}UI{PURPLE}G{BLACK}R{2SPACE}
    B{DOWN,9LEFT}G{SPACE,GREEN}JK{BLACK,SP
    ACE}R N";                                 <176>
312 PRINT"{DOWN,8LEFT}G{SPACE,PURPLE}R{BLA
    CK,2SPACE}RN{DOWN,7LEFT}TTTTTT{HOME}";
    :RETURN                                   <062>
313 PRINT"{HOME,4DOWN,4RIGHT,BLACK}NM{DOWN
    ,3LEFT}N{2SPACE}M{DOWN,4LEFT,RVSON}*{R
    VOFF,3SPACE}M{DOWN,5LEFT,RVSON,SPACE}*
    {RVOFF,3SPACE}MGGGGGG{DOWN,12LEFT}*{RV
    SON,SPACE}*{RVOFF,2SPACE}NM";             <036>
314 PRINT"{YELLOW,RVSON}VVVVV{RVOFF,BLACK}
    NG{DOWN,12LEFT}*{RVSON,SPACE}*{RVOFF}N
    {YELLOW,RVSON}VVVVV{RVOFF,BLACK}N G{DO
    WN,11LEFT}*£QTTTTP{2SPACE}G";             <215>
315 PRINT"{DOWN,9LEFT}G{SPACE,GREEN}UI{SPA
    CE,BLACK}R{2SPACE}B{DOWN,9LEFT}G{SPACE
    ,GREEN}JK{BLACK,SPACE}R N{DOWN,8LEFT}G
    {4SPACE}RN{DOWN,7LEFT}TTTTTT{HOME}"       <100>
316 RETURN                                    <120>
317 PRINT S$TAB(4)"{9UP,RED}NTTTTTTTTTTTM
    {7RIGHT}N{14SPACE}M{6RIGHT}LGGGGGGGGGG
    GGGGS";                                   <160>
318 PRINT"{6RIGHT,BLUE}M{GREEN}V+V+V+V+V+V
    +V+{BLUE}N{7RIGHT}M{PURPLE}ZZZZZZZZZZZ
    Z{BLUE}N{9RIGHT}M{RVSON}£*£*£*£*£*{RVO
    FF}N";                                    <242>
319 PRINT"{10RIGHT,RVSON}£{10SPACE}*{HOME}
    ":RETURN                                  <198>
320 PRINT"{HOME}"TAB(14)"{RED}U**I{DOWN,4L
    EFT}-UI-{DOWN,4LEFT}JK{RVSON}TT{RVOFF}
    ";:FOR A=1 TO 5:PRINT"{DOWN,2LEFT}--{D
    OWN,2LEFT,RVSON}TT{RVOFF}";:NEXT          <125>
321 PRINT"{DOWN,2LEFT}JK{HOME}":RETURN        <124>
322 PRINT S$TAB(7)"{4UP,BLACK,RVSON}£GUTY
    R*{DOWN,9LEFT}£ I{4SPACE}M *{RVOFF}";     <134>
323 PRINT"{DOWN,10LEFT}Y TU YYUTT{HOME}":R
    ETURN                                     <172>
324 PRINT S$TAB(4)"{4UP,BLACK,RVSON}£G{10S
    PACE}R*{7RIGHT}£ G{10SPACE}R *";          <159>
325 PRINT"{5RIGHT}£{2SPACE}G{10SPACE}R{2SP
    ACE}*{HOME}":RETURN                       <220>
326 PRINT"{HOME}"TAB(7)"{RED}G{2RIGHT}G{DO
    WN,5LEFT,GREEN,RVSON}£{RVOFF,YELLOW}B{
    RVSON,GREEN}*£{RVOFF,YELLOW}B{RVSON,GR
    EEN}*{6RIGHT}£{DOWN,13LEFT,6SPACE,3RIG
    HT}£{2SPACE,RVOFF}£";                     <114>
327 PRINT"{DOWN,14LEFT,RVSON}£{RVOFF}£{BLA
    CK}VVVV{GREEN}*{RVSON}*£{SPACE,RVOFF}£
    {DOWN,12LEFT,RVSON}£{SPACE,RVOFF,6SPAC
    E,RVSON,SPACE}M{RVOFF}£{DOWN,11LEFT}";    <176>
328 PRINT"{RVSON,2SPACE,RVOFF,6SPACE,RVSON
    ,2SPACE,DOWN,10LEFT,2SPACE,RVOFF,6SPAC
    E,RVSON,2SPACE,DOWN,10LEFT,2SPACE,RVOF
    F,2SPACE,BLACK,RVSON}*£{RVOFF,2SPACE,G
    REEN,RVSON,2SPACE}G*{DOWN,12LEFT}";       <045>
329 PRINT"{2SPACE}*{RVOFF,SPACE,RVSON,BLAC
    K}RG{RVOFF,SPACE,RVSON,GREEN}£{2SPACE}
    G *{DOWN,13LEFT,RVOFF}*{RVSON,2SPACE}*
    {BLACK}RG{GREEN}£{2SPACE,RVOFF}£*{RVSO
    N,2SPACE}*{RVOFF,DOWN,13LEFT}";           <058>
330 PRINT"*{RVSON,6SPACE,RVOFF}£{2RIGHT}*{
    RVSON,2SPACE,DOWN,12LEFT,RVOFF}*{RVSON
    ,4SPACE,RVOFF}£{3RIGHT,RVSON}£{2SPACE,
    DOWN,11LEFT,4SPACE,3RIGHT}£{2SPACE,RVO
    FF}£";                                    <000>
331 PRINT"{DOWN,11LEFT,RVSON,4SPACE,2RIGHT
    }£{2SPACE,RVOFF}£"SPC(12)"*{RVSON,3SPA
    CE}*£{2SPACE,RVOFF}£"SPC(14)"*{RVSON,3
    SPACE}M{SPACE,RVOFF}£"                    <148>
332 PRINT S$"{BLACK}EINE RIESIGE KOBRA BE-
    DROHT DICH. WAS WILLSTDU MACHEN ???":R
    ETURN                                     <179>
333 PRINT S$TAB(8)"{7UP,BLACK}U*CC*I{DOWN,
    6LEFT}-{SPACE,RED}MN{SPACE,BLACK}-{DOW
    N,9LEFT}U**IUFFIU**I{DOWN,12LEFT}";       <117>
334 PRINT"-U*IT{GREEN}BB{BLACK}GU*I-{DOWN,
    12LEFT}-- --{BLUE}RG{BLACK}-- --{DOWN,
    12LEFT}K- NJ**KM -J";                     <075>
335 PRINT"{DOWN,11LEFT}KR TEET GJ{6RIGHT,B
    LACK}EINE{3SPACE}GGEFAEHRLICHE TARANT
    EL VERSPERRT DIR";                        <135>
336 PRINT"DEN WEG ...":GOTO 44                <003>
337 PRINT"{HOME}"TAB(10)"{6DOWN,BLACK}UI{D
    OWN,2LEFT}JK":PRINT S$"IN DER WAND IST
     EIN{3SPACE}LOCH, IN DAS GENAU"           <142>
338 PRINT"EINE DYNAMITSTANGE{4SPACE}PASSEN
     WUERDE (WAS{4SPACE}FUER EIN ZUFALL) .
    ..";:RETURN                               <147>
339 PRINT"{HOME}"TAB(9)"{DOWN,CYAN}U**I{DO
    WN,4LEFT}-NN-{DOWN,4LEFT}-NN-{DOWN,4LE
    FT}JIUK{DOWN,5LEFT}U*KJ*I{DOWN,6LEFT}-
    NNNN-";                                   <071>
340 PRINT"{DOWN,6LEFT}-NNNN-{DOWN,6LEFT}--
    NN--{DOWN,6LEFT}--NN--{DOWN,6LEFT}JKNN
    JK{DOWN,5LEFT}-UI-";                      <040>
341 PRINT"{DOWN,4LEFT}----{DOWN,5LEFT}UK--
    JI{DOWN,6LEFT}J*KJ*K"                     <242>
342 PRINT S$"{BLACK}DU HAST DIE MOERDER-{2
    SPACE}MUMIE AUFERWECKT. SIE WILL DICH
    JETZT UM-"                                <078>
343 PRINT"BRINGEN !!!":RETURN                 <051>
344 POKE Q,254:PRINT"{CLR,YELLOW}"TAB(12)"
    **{2SPACE,RVSON,6SPACE}";:PRINT SPC(16
    )"{6SPACE,RVOFF}";                        <245>
345 PRINT"{SPACE,CYAN}GPOIOPG{6SPACE,YELLO
    W}N *{RVSON,5SPACE,RVOFF,SPACE,CYAN}TY
    U{RVSON}T{RVOFF}UYT{5SPACE,YELLOW}N{3S
    PACE}*{RVSON,4SPACE,RVOFF}";              <175>
346 PRINT"{CYAN,6SPACE}GPOIOPG{3SPACE,YELL
    OW}N *{RVSON,3SPACE,RVOFF,6SPACE,CYAN}
    TYU{RVSON}T{RVOFF}UYT{2SPACE,YELLOW}N{
    6SPACE,CYAN}GPOIOPG{12SPACE,YELLOW}-"     <073>
347 PRINT"{CYAN}TYU{RVSON}T{RVOFF}UYT{10SP
    ACE,BLACK,RVSON}£*":PRINT TAB(16)"{RVS
    ON}£{2SPACE}*":PRINT TAB(15)"{RVSON}£{
    4SPACE}*"                                 <223>
348 PRINT TAB(14)"{RVSON}£{6SPACE}*{RVOFF}
    GGRRFF**CCDDEETYU{RVSON}TT{RVOFF}UYT{D
    OWN}"                                     <159>
349 FOR A=1 TO 22:PRINT"{RED}={BLUE}";:NEX
    T:GOTO 44                                 <020>
350 POKE Q,254:PRINT"{CLR,BLACK,DOWN,2SPAC
    E,RVSON}£ *£ *{SPACE,RIGHT,SPACE}£ *{2
    SPACE}*{3SPACE,4RIGHT,SPACE,2RIGHT,SPA
    CE,2RIGHT,SPACE,RIGHT,SPACE}G{RIGHT}R{
    SPACE,RIGHT}R "                           <157>
351 PRINT"{2SPACE}*{RVSON,SPACE}*{SPACE,2R
    IGHT,3SPACE}G R{SPACE,RIGHT}R{2SPACE,7
    RIGHT}R{SPACE,2RIGHT,SPACE,RIGHT,SPACE
    }G{RIGHT}R{SPACE,RIGHT}R "                <068>
352 PRINT"{2SPACE}*{RVSON,SPACE,RVOFF}£*{R
    VSON,SPACE,RVOFF}£{RVSON,SPACE,RIGHT,S
    PACE}G{RIGHT}R{2SPACE,RVOFF}£{RVSON,3S
    PACE}"                                    <009>
353 PRINT"{DOWN,BLUE}DU BIST LEIDER TOD,"     <049>
354 ON W GOSUB 356,358,359,360,361,363,365
    ,366,367                                  <055>
355 PRINT"{DOWN}":GOTO 380                    <230>
356 PRINT"{DOWN}WEIL DICH DIE MUMIE{3SPACE
    ,DOWN}MIT IN DIE EWIGEN JAGD{DOWN}GRUE
```

Listing »DANGEROUS PYRAMID« (Fortsetzung)

```
     NDE GENOMMEN HAT !"                       <122>
357 RETURN                                     <161>
358 PRINT"{DOWN}WEIL DIE TARANTEL DICH{DOW
     N}GEBISSEN HAT !!!":RETURN                <006>
359 PRINT"{DOWN}WEIL DIE SCHLANGE DICH{DOW
     N}VERSPEISST HAT !!!":RETURN              <168>
360 PRINT"{DOWN}WEIL DU ABGESTUERZT{3SPACE
     ,DOWN}BIST !!!":RETURN                    <245>
361 PRINT"{DOWN}WEIL DU MIT DER FACKEL{DOW
     N}ODER MIT EINEM STREICH{DOWN}HOLZ DAS
      DYNAMIT ANGE-"                           <067>
362 PRINT"ZUENDET HAST UND DABEI{DOWN}EXPL
     ODIERT BIST !!!":RETURN                   <140>
363 PRINT"{DOWN}WEIL DU DIR AN EINEM{2SPAC
     E,DOWN}GEGENSTAND, DEN DU{4SPACE,DOWN}
     NEHMEN WOLLTEST, EINEN"                   <240>
364 PRINT"BRUCH GEHOBEN HAST !!":RETURN     <076>
365 PRINT"{DOWN}WEIL DU DICH DURCH DAS{DOW
     N}TRINKEN DES GIFTES{4SPACE,DOWN}SELBS
     T VERGIFTET HAST":RETURN                  <181>
366 PRINT"{DOWN}WEIL DU DICH GE-{6SPACE,DO
     WN}SCHNITTEN ODER VER-{3SPACE,DOWN}BRA
     NNT HAST !!!":RETURN                      <160>
367 PRINT"{DOWN}WEIL DU AN DIE DECKE{2SPAC
     E,DOWN}GESPRUNGEN BIST UND{3SPACE,DOWN
     }DIR DEN SCHAEDEL EIN-"                   <003>
368 PRINT"{DOWN}GESCHLAGEN HAST !!!":RETUR
     N                                         <103>
369 POKE Q,250:AZ=62:GOSUB 370:GOTO 373     <021>
370 FOR A=2 TO 5:POKE Q-A,0:NEXT:POKE Q-1,
     8                                         <106>
371 FOR A=1 TO AZ:READ T,L,R:POKE Q-R,163+
     T:FOR B=0 TO L*175:NEXT B,A               <148>
372 FOR A=8 TO 0 STEP-.05:POKE Q-1,A:NEXT:
     FOR A=2 TO 5:POKE Q-A,0:NEXT:RETURN       <065>
373 POKE Q,186:PRINT"{CLR,BLACK,2SPACE}@@
     @@{2SPACE}@ @ @ @{3SPACE}@{3SPACE}N NV
      NMNNVNVNVNM NN@"                          <028>
374 PRINT"{SPACE,RVSON,2SPACE}*{2SPACE}*£
     *{SPACE,RIGHT,SPACE}£ *{RVOFF}N@{RVSON
     ,SPACE,RVOFF,SPACE}@{2SPACE,RVSON,SPAC
     E,RVOFF}N{RVSON,2SPACE,RVOFF}N{RVSON,2
     SPACE,RVOFF}N{RVSON,2SPACE,RIGHT,2SPAC
     E,RIGHT,SPACE,RVOFF,SPACE}@{RVSON,SPAC
     E,RVOFF,SPACE}@"                          <195>
375 PRINT"{SPACE,RVSON,5SPACE,RVOFF}£{RVSO
     N,4SPACE,RIGHT,2SPACE,RIGHT,SPACE,RVOF
     F,SPACE}@{RVSON,SPACE,RVOFF,SPACE}@{2S
     PACE,RVSON,SPACE,RVOFF}N{RVSON,2SPACE,
     RVOFF}*{RVSON}*{SPACE,RIGHT,2SPACE,RVO
     FF}N{RVSON,2SPACE,RVOFF}N{RVSON,SPACE,
     RVOFF}N{SPACE,RVSON,SPACE,RVOFF}N@"       <093>
376 PRINT"{SPACE,RVSON,2SPACE,RVOFF}£{RVSO
     N,SPACE,RVOFF}N{RVSON,2SPACE,RVOFF}N{R
     VSON,SPACE,RVOFF}*{RVSON,SPACE,RVOFF}£
     *{RVSON,SPACE,RVOFF}£{2SPACE,RVSON}T{R
     VOFF}N"                                   <120>
377 PRINT"{DOWN,BLUE}DU HAST DAS{SPACE,RVS
     ON}ADVENTURE{RVOFF,SPACE,DOWN}TATSAECH
     LICH GELOEST,{SPACE,DOWN}TROTZ DEN VIE
     LEN"                                      <082>
378 PRINT"{DOWN}SCHWIERIGKEITEN UND{3SPACE
     ,DOWN}FALLEN !!!!!"                       <089>
379 PRINT"{DOWN,PURPLE}HERZLICHEN{12SPACE}
     GLUECKWUNSCH"                             <097>
380 PRINT"{DOWN,RED}WILLST DU DICH NOCH{3S
     PACE}EINMAL VERSUCHEN ???{HOME}"          <127>
381 GET A$:IF A$="J"THEN RUN                <053>
382 IF A$="N"THEN POKE 0,0:SYS 0            <036>
383 GOTO 381                                <233>
384 POKE Q-2,140:FOR A=15 TO 0 STEP-.08:PO
     KE Q,41:POKE Q-1,A:POKE Q,121:NEXT        <136>
385 IF Z=1 THEN Z=0:POKE Q-2,0:RETURN       <170>
386 FOR A=1 TO 1000:NEXT:POKE Q-2,0         <003>
387 GOTO 391                                <253>
388 POKE Q,154                              <189>
389 FOR C=1 TO 3:POKE Q-2,150:FOR D=0 TO 1
     5 STEP.3:POKE Q-1,D:NEXT D:POKE Q-2,0     <240>
390 POKE Q,42:FOR E=1 TO 100:NEXT E:POKE Q
     ,154:NEXT C:FOR F=1 TO 1000:NEXT F        <196>
391 POKE Q-1,15                             <041>
392 A=225:B=4:GOSUB 397:B=3:GOSUB 397:B=1:
     GOSUB 397:B=4:GOSUB 397                   <009>
393 A=229:B=3:GOSUB 397:A=228:B=1:GOSUB 39
     7:B=3:GOSUB 397                           <107>
394 A=225:B=1:GOSUB 397:B=3:GOSUB 397:A=22
     3:B=1:GOSUB 397                           <051>
395 A=225:B=6:GOSUB 397                     <113>
396 GOTO 350                                <166>
397 POKE Q-5,A:POKE Q-4,A:POKE Q,24:FOR C=
     1 TO B*200:NEXT:POKE Q,25:POKE Q-4,0      <053>
398 POKE Q-5,0:FOR C=1 TO 20:NEXT:RETURN    <192>
399 PRINT"{CLR}":POKE Q,40:POKE Q-1,15:B=2
     3:FOR A=220 TO 130 STEP-1:POKE Q-3,A:B
     =B-.25                                    <010>
400 IF B=INT(B)THEN GOSUB 402               <118>
401 FOR C=1 TO 5:NEXT C,A:GOTO 403          <225>
402 POKE Q-15,35-B:POKE Q-14,82-INT(B/2)*4
     :POKE Q-13,B-1:POKE Q-12,2*B:RETURN       <249>
403 POKE Q-15,12:POKE Q-14,38:POKE Q-13,22
     :POKE Q-12,46:POKE Q-3,0:POKE Q-2,130     <250>
404 PRINT"{HOME,6DOWN}"SPC(8)"{BLACK}@{4SP
     ACE}@"SPC(15)"MM{3SPACE}MM"SPC(16)"MM{
     2SPACE}NN"SPC(17)"UCCI"                   <150>
405 PRINT SPC(9)"_{GREEN}88{BLACK}_"SPC(18
     )"_{BLUE}££{BLACK}_"SPC(18)"JFFK"SPC(1
     7)"NN{2SPACE}MM"SPC(15)"MM{3SPACE}MM"     <160>
406 PRINT SPC(8)"T{4SPACE}T":FOR A=15 TO 0
      STEP-.1:POKE Q-15,11:POKE Q-1,A           <135>
407 FOR B=1 TO 10:NEXT B:POKE Q-15,12:FOR
     B=1 TO 10:NEXT B,A:POKE Q-2,0             <232>
408 FOR A=1 TO 1500:NEXT:GOTO 391           <231>
409 END                                     <157>
410 POKE 198,0                              <064>
411 POKE 4601,160:POKE 38393,2:FOR A=1 TO
     350:NEXT                                  <080>
412 GET C$:IF C$=""THEN POKE 4601,32:FOR A
     =1 TO 350:NEXT:GOTO 411                   <101>
413 POKE 4601,32:RETURN                     <092>
414 FOR A=4096+16*22 TO 4096+23*22:POKE A,
     32:NEXT:RETURN                            <133>
415 DATA NSOW.R,.S.....,.S.W..,.SO...,N..W.
     .,NSO...,NS......,..WH.,N.O...,N....R     <111>
416 DATA ...W.R,..OW...,SOW..,.SO......,..W.
     .,.SOW..,N.O...,NS......,.S.W...,N.OW..   <038>
417 DATA .SOWH.,NSO...,N..W...,..OW..,N.OW.
     .,N.O....,.S..H.,.S.W...,..O....,.S.W..   <187>
418 DATA .SO....,N..W...,..OW...,..OW..,NSO..
     .,NS.....,.S.W...,..O......,..W..,N.O...  <120>
419 DATA NS.....,NS.W...,.SO.......,.W...,.SO..
     .,NS.....,N......,N..W...,..OW..,N.OW..   <021>
420 DATA N.O...                             <193>
421 :                                       <143>
422 DATA SCHATZTRUHE,4,SARKOPHAG,39,SCHATZ
     TRUHE,0,KUHLE,36,FALLGRUBE,46,PHARAONE
     NSTAB                                     <095>
423 DATA 0,DYNAMIT,5,DOLCH,12,FACKEL,14,SA
     CK,16,SCHLUESSEL,22,STREICHHOELZER,25     <078>
424 DATA BRETT,29,ZAUBERTRANK,28,MASKE,38,
     GIFT,51,DIAMANT,19                        <063>
425 :                                       <147>
426 DATA MUMIE,0,TARANTEL,50,KOBRA,43,LOCH
      IN WAND,33                                <239>
427 :                                       <149>
428 DATA-5,5,-1,1,-4,4,-1,1,-3,3,-1,1       <087>
429 :                                       <151>
430 DATA GEHE,LEGE,NIMM,WIRF,OEFF,ZUEN,STE
     C,TRIN,SAGE,SPRI,BESI,ENDE,SCHL,KLET      <158>
431 DATA SAVE,LOAD                          <175>
432 :                                       <154>
433 DATA 32,2,3,0,0,4,46,2,3,-40,0,3,46,1,
     3,52,1,3,56,2,3,46,2,3,32,0,4,62,4,3,0
     ,0,4                                      <217>
434 DATA 56,3,3,-40,0,3,56,1,3,24,0,4,58,2
     ,3,62,1,3,58,1,3,0,0,4,56,1,3,58,1,3,6
     2,2,3                                     <165>
435 DATA 32,0,4,52,1,3,46,1,3,52,1,3,56,1,
     3,52,2,3,-40,0,4,32,2,3,0,0,4,46,2,3,-
     40,0,3                                    <195>
436 DATA 46,1,3,52,1,3,56,2,3,46,2,3,32,0,
     4,62,3,3,58,1,3,0,0,4,56,3,3,-40,0,3,5
     6,1,3                                     <100>
437 DATA 24,0,4,58,1.1,3,62,1.2,3,0,0,4,56
     ,1.3,3,58,1.4,3,32,0,4,52,.2,3,56,.2,3    <213>
438 DATA 52,.2,3,56,.2,3,52,.2,3,56,.2,3,5
     2,3.2,3,46,1.6,3,-40,0,3,0,0,4,46,6,3     <139>
```

Listing »DANGEROUS PYRAMID« (Schluß)

Checksummer 20 V3

Der Checksummer 20 V3 für den VC 20 überprüft jede Basic-Zeile direkt nach der Eingabe, erkennt Fehleingaben und auch Vertauschungen von Zahlen und Ziffern, und erspart Ihnen deshalb eine aufwendige Fehlersuche.

Der Checksummer 20 V3 ist ein kleines Maschinenprogramm, das Sie sofort unterrichtet, ob Sie die jeweilige Programmzeile korrekt eingegeben haben.

So gehen Sie vor:

1. Programm abtippen und speichern.
2. Starten mit RUN.
3. Anschalten des Checksummer 20 V3 mit SYS 981.
4. Test: Geben Sie in einer freien Zeile ein: »1 REM« und drücken die RETURN-Taste. Am Bildschirm oben links sollten Sie die Prüfsumme <63> sehen.
5. Geben Sie ein Listing aus unserem Heft ein. Nach jeder Zeile wird die Zahl, die im Listing in Klammern < > steht, in den Bildschirm eingeblendet. Stimmen die Zahlen nicht überein, so liegt vermutlich ein Eingabefehler vor.
Die Zahl in den Klammern und auch die Klammern selbst dürfen beim Abtippen nicht mit eingegeben werden!
6. Dieser Checksummer 20 V3 bemerkt auch Vertauschungen von Zahlen und Buchstaben.
7. Abschaltung wird mit »SYS 58459« vollzogen.

Achtung: Nehmen Sie keine Kassetten-Operationen vor, wenn der Checksummer VC 20 eingeschaltet ist. Da das Betriebssystem den Kassettenpuffer mit Daten belegt, kann der Checksummer VC 20 überschrieben werden. Wollen Sie deshalb ein Programm auf (von) Kassette speichern (laden), so müssen Sie erst den Checksummer VC 20 abschalten.

Als Sicherung wird bei der Initialisierung geprüft, ob das zuletzt angesprochene Peripherie-Gerät der Kassettenrecorder war. Ist das der Fall, so werden die Betriebssystemroutinen LOAD und SAVE für die Benutzung gesperrt. Der Computer meldet bei Aufruf einer dieser beiden Routinen READY, ohne weitere Aktionen durchzuführen. Diese Sicherung kann man nach der Tipparbeit aufheben, wenn man den Checksummer VC 20 mit SYS 58459 abschaltet. Weiterhin wird dann durch gleichzeitiges Drücken der Tasten »Run-Stop & Restore« erreicht, daß die Betriebssystemroutinen LOAD und SAVE wieder eingerichtet werden.

– Bei Benutzung einer Diskettenstation brauchen Sie nicht darauf zu achten, daß bei LOAD beziehungsweise SAVE der Checksummer VC 20 überschrieben wird, da der Kassettenpuffer für die Diskettenstation normalerweise nicht genutzt wird. Bitte beachten Sie auch die folgende Seite.

(F. Lonczewski/gk)

```
5 PRINTCHR$(14)
10 PRINT"▯"
20 PRINT"├──────────────────────────┤"
30 PRINT"▓▓▓▓  TEST ▒        "
40 PRINT"▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀"
```

Bild 2. Auf dem Bildschirm oder Ihrem Drucker sieht das Listing (Bild 1) so aus

```
10 REM***************
11 REM*             *
12 REM* CHECKSUMMER *
13 REM*             *
14 REM* V3     VC20 *
15 REM*             *
16 REM*   WRITTEN   *
17 REM* MAERZ  1985 *
18 REM*      BY     *
19 REM*F. LONCZEWSKI*
20 REM***************
21 PRINT"{CLR,SPACE,RVSON}CHECKSUMMER V3 V
   C-20{RVOFF}"
22 PRINT"{2DOWN}EINEN MOMENT, BITTE..."
23 FOR I=827 TO 1019:READ A:POKE I,A
24 PS=PS+A+1:NEXT I
25 IF PS<>24464 THEN PRINT"{DOWN}PRUEFSUMM
   ENFEHLER !":END
26 SYS 981:PRINT"CHECKSUMMER AKTIVIERT."
27 PRINT"AN :SYS981"
28 PRINT"{DOWN}AUS:SYS58459, BEI CAS-{4SPA
   CE}SETTE ZUSAETZLICH{5SPACE}RUN/STOP &
   RESTORE"
29 PRINT"{DOWN}BEI AKTIVIERTEM CHECK-SUMME
   R KEIN";
30 PRINT" CASSETTEN-BETRIEB (LOAD, SAVE){2
   SPACE}ERLAUBT!":NEW
31 DATA 32,95,3,134,122,132,123,32,115,0,1
   70,240,243,162,255
32 DATA 134,58,144,10,162,0,134,255,32,121
   ,197,76,225,199,162
33 DATA 1,134,255,76,156,196,166,255,224,1
   ,240,3,76,96,197
34 DATA 160,2,169,0,170,133,254,177,95,240
   ,40,201,32,208,3
35 DATA 200,208,245,133,253,138,41,7,170,2
   40,14,72,165,253,24
36 DATA 42,105,0,202,208,249,133,253,104,1
   70,232,165,253,24,101
37 DATA 254,133,254,76,119,3,192,4,48,219,
   198,214,165,214,72
38 DATA 162,3,169,32,157,1,4,189,209,3,32,
   210,255,202,16
39 DATA 242,166,254,169,0,32,205,221,169,6
   2,32,210,255,104,133
40 DATA 214,32,135,229,169,141,32,210,255,
   162,0,134,255,240,148
41 DATA 9,60,18,19,169,59,141,2,3,169,3,14
   1,3,3,165
42 DATA 186,201,1,208,16,169,116,141,48,3,
   141,50,3,169,196
43 DATA 141,49,3,141,51,3,173,136,2,141,17
   0,3,96
```

Listing. Checksummer VC 20 V3

```
5 PRINT CHR$(14)                                <242>
10 PRINT"{CLR}"                                 <254>
20 PRINT"Q****************************W"        <130>
30 PRINT"{4DOWN,2SPACE}TEST{SPACE,BLUE,6SP
   ACE}"                                        <022>
40 PRINT"BBBBBBBBBBBBBBBBBBBBBBBBBBBBB"         <108>

© 64'er
```

Bild 1. So könnte ein Teil eines Listings abgedruckt sein. In Zeile 10 müssen Sie nach den Anführungsstrichen die CLEAR/HOME-Taste drücken und nicht die Klammern mit dem Wort CLR. In Zeile 20 drücken Sie nach den Anführungsstrichen die Commodore-Taste und den Buchstaben Q, gefolgt von mehreren SHIFT- und Stern-Tasten und zum Schluß die Commodore-Taste und den Buchstaben W. In Zeile 30 ist es viermal die Cursor-nach-unten-Taste, gefolgt von zweimal die Leertaste, dann SHIFT und T und normal EST, zum Schluß noch einmal die Leertaste, die Farbtaste Blau (Control und 7) und sechsmal die Leertaste. Zeile 40 besteht lediglich aus mehreren Grafikzeichen, die mit der Commodore-Taste und <B> erzeugt werden. (gk)

Wie unsere Basic-Programme einzugeben sind

Sie haben kein Problem mehr mit dem Abtippen von Basic-Programmen, wenn Sie die folgenden Hinweise zum Abtippen beachten.

Unsere Basic-Listings enthalten keine Steuerzeichen mehr. Diese werden ersetzt durch Klartext und stehen zwischen geschweiften Klammern. Deshalb sind weder die Klammern noch was dazwischen steht, abzutippen, sondern die in Tabelle 1 aufgeführten Tasten zu drücken. Auf Ihrem Bildschirm erhalten Sie dann wieder die entsprechenden Grafikzeichen (siehe Bild 1 und 2 auf Seite 129).

Alle Grafikzeichen werden ebenfalls ersetzt durch unterstrichene oder überstrichene Großbuchstaben.

Unterstrichene Buchstaben bedeuten, daß Sie die SHIFT-Taste und den angegebenen Buchstaben drücken müssen, überstrichene jedoch die Commodore-Taste mit dem Buchstaben.

Auch hier erhalten Sie am Bildschirm das entsprechende Grafikzeichen und nicht etwa das im Listing erkennbare Zeichen (siehe Bild 1 und 2 auf Seite 129).

Die Leerzeichen zwischen den einzelnen Basic-Befehlen können beim Abtippen entfallen. Dies ist besonders bei speicherkritischen Programmen wichtig.

Ebenso müssen Zeilen, die mehr als 80 Zeichen pro Zeile enthalten, mit den bekannten Abkürzungen für die Basic-Befehle (siehe auch das Handbuch zum C 16, Seite 202) eingegeben werden. Die in den eckigen Klammern (< . . . >) stehenden Zahlen am Ende jeder Basic-Zeile sind Prüfsummen. Sie werden benötigt, wenn Sie den Checksummer (siehe Seite 129) verwenden. Dieser Checksummer ist jedoch nur für den VC 20 gültig. Deshalb wurden auch nur VC 20-Listings mit Prüfsummen versehen. Selbstverständlich können Sie die Basic-Listings auch ohne den Checksummer eingeben. Dann sind die Prüfsummen für Sie ohne Bedeutung.

Programme für den C 16 sind auch gültig für den C 116. Auch wenn Sie Plus/4-Besitzer sind, dürfte es kaum Probleme geben. VC 20-Listings können C 16-, C 116- und Plus/4-Besitzer nur bedingt verwenden: Wenn Sie in einem VC 20-Listing POKE-Befehle finden, läuft das Programm auf dem C 16, C 116 oder Plus/4 mit großer Wahrscheinlichkeit nicht. (gk)

Steuerzeichen	Bedeutung
{CTRL}	steht für Control-Taste, so bedeutet {CTRL-A}, daß Sie die Control-Taste und die Taste »A« drücken müssen. Im folgenden steht:
{DOWN}	Taste neben rechtem Shift, Cursor unten
{UP}	Shift-Taste & Taste neben rechtem Shift; Cursor hoch
{CLR}	Shift-Taste & 2. Taste ganz rechts oben
{INST}	Shift-Taste & Taste ganz rechts oben
{HOME}	2. Taste von ganz rechts oben
{DEL}	Taste ganz rechts oben
{RIGHT}	Taste ganz rechts unten
{LEFT}	Shift-Taste & Taste unten rechts
{SPACE}	Leertaste, Hinweis: {13 SPACE} bedeutet 13mal die Leertaste drücken
{SHFT-SPCE}	Shift-Taste & Leertaste
{F1}	grauer Tastenblock rechts
{F3}	grauer Tastenblock rechts
{F5}	grauer Tastenblock rechts
{F7}	grauer Tastenblock rechts
{F2}	grauer Tastenblock rechts & Shift
{F4}	grauer Tastenblock rechts & Shift
{F6}	grauer Tastenblock rechts & Shift
{F8}	grauer Tastenblock rechts & Shift
{RETURN}	Shift-Taste & Return

Steuerzeichen	Bedeutung
{CTRL}	Control-Taste
{BLACK}	Control-Taste & 1
{WHITE}	Control-Taste & 2
{RED}	Control-Taste & 3
{CYAN}	Control-Taste & 4
{PURPLE}	Control-Taste & 5
{GREEN}	Control-Taste & 6
{BLUE}	Control-Taste & 7
{YELLOW}	Control-Taste & 8
{RVSON}	Control-Taste & 9
{RVOFF}	Control-Taste & 0
{ORANGE}	Commodore-Taste & 1
{BROWN}	Commodore-Taste & 2
{LIG.RED}	Commodore-Taste & 3
{GREY 1}	Commodore-Taste & 4
{GREY 2}	Commodore-Taste & 5
{LIG.GREEN}	Commodore-Taste & 6
{LIG.BLUE}	Commodore-Taste & 7
{GREY 3}	Commodore-Taste & 8

Wenn Sie sich erst einmal an die in Klartext geschriebenen Steuerzeichen gewöhnt haben, werden Sie den Vorteil dieser Schreibweise erkennen. Der zu dem jeweiligen Steuerzeichen gehörende Klartext ist so verfaßt, daß Sie leicht die Taste beziehungsweise die Tastenkombination finden, die Sie drücken müssen.

Tabelle 1. Die Steuerbefehle für den C64/VC20/C16 im Klartext

Directory einmal anders

Wenn Sie der schwache Informationsgehalt des Directory einer Diskette stört, dann schafft das folgende Programm Abhilfe. Es druckt ein Inhaltsverzeichnis mit allen wichtigen Informationen auf dem Drucker aus.

Directory-List arbeitet auf dem C16, C116 und Plus/4 und erlaubt das Ausdrucken von sehr komfortablen Directories. Dieser Ausdruck enthält Filetyp und -länge, sowie Startadresse eines Programms im Speicher, Starttrack und -sektor auf der Diskette und den Block, unter dem es im Directory gespeichert ist (Bild 1 zeigt einen Beispielausdruck).

Listing 1 enthält das Programm zu »Directory-List«. Da wir für den C 16 und den Plus/4 leider noch keinen Checksummer anbieten können, wurde das Listing ohne Prüfsummen abgedruckt.

Die Handhabung ist denkbar einfach: Nach dem Starten mit RUN kann die gewünschte Diskette in das Laufwerk eingelegt werden. Anschließend wird der Ausdruck durch einen Tastendruck gestartet. Das Programm meldet sich nach Beendigung des Druckvorgangs mit dem Status der Floppystation und muß bei weiteren Ausdrucken erneut mit RUN gestartet werden. (S.Giero/ks)

Name: ID: Format:

Filename:	Filetyp:	Bloecke:	Startadresse:	Spur:	Sektor:	Dirblock:
how to use	prg	14	$0801/ 2049	17	0	18 1
how part two	prg	8	$0801/ 2049	17	5	18 1
how part three	prg	7	$0801/ 2049	19	0	18 1
vic-20 wedge	prg	4	$0401/ 1025	19	4	18 1
c-64 wedge	prg	1	$0401/ 1025	19		
dos 5.1	---		--		0	18 7
	prg	8	$0401/ 1025	12	19	18 7
unscratch	prg	7	$0401/ 1025	24	1	18 10
header change	prg	5	$0401/ 1025	24	10	18 10

Bloecke: insgesamt = 664 belegt = 261 frei = 403

Bild 1. Beispielausdruck eines erweiterten Directory

```
80 COLOR 0,1: COLOR 4,2,2: COLOR 1,14: SCNCLR
90 T=18: S=1
100 FOR I=1 TO 4: READ F$(I): NEXT
110 DATA SEQ,PRG,USR,REL
150 PRINT "{RVSON}DIRECTORY LIST{2SPACE}BY SVEN
    GIERO, BURGDORF ": PRINT
160 PRINT "DIESES PROGRAMM DRUCKT DAS DIRECTORY"
170 PRINT "EINER DISKETTE MIT ALLEN DAZUGEHOERIG
    EN"
180 PRINT "DATEN ( SPUR, SEKTOR, STARTADRESSE US
    W )"
190 PRINT "DER EINZELNEN FILES AUF DEM DRUCKER A
    US.": PRINT
200 PRINT "BITTE SCHALTEN SIE DEN DRUCKER AN, LE
    GEN"
210 PRINT "DIE ENTSPRECHENDE DISKETTE INS LAUFWE
    RK"
220 PRINT "UND DRUECKEN SIE EINE TASTE !"
230 GET KEY A$: SCNCLR
240 POKE 65286,PEEK(65286) AND 239
290 OPEN 1,8,15,"I0": IF DS THEN 790
300 OPEN 2,8,2,"#"
310 OPEN 4,4,7: PRINT#4
320 PRINT#1,"U1 2 0 18 0"
330 PRINT#1,"B-P 2 144"
340 FOR I=1 TO 16
350 GET #2,A$: N$=N$+A$
360 NEXT
370 GET #2,A$,A$,A$,B$: I$=A$+B$
380 GET #2,A$,A$,B$: F$=A$+B$
390 PRINT#4,"NAME:{RVSON}"N$"{RVOFF} ID:{RVSON}"
    I$"{RVOFF} FORMAT:{RVSON}"F$: PRINT#4
400 PRINT#4,"FILENAME:" SPC(9)"FILETYP: BLOECKE:
    ";
410 PRINT#4,"STARTADRESSE: SPUR: SEKTOR: DIRBLOC
    K:"
420 FOR I=1 TO 80: PRINT#4,"=";: NEXT : PRINT#4
470 IF T=0 THEN 760
480 PRINT#1,"U1 2 0";T;S
490 Z=2: T1=T: S1=S
500 GET #2,A$,B$: T=ASC(A$): S=ASC(B$)
510 IF Z>256 THEN 470
520 PRINT#1,"B-P 2";Z: N$=""
530 GET #2,A$,B$,C$
540 FT=ASC(A$) AND 7: ET=ASC(B$): ES=ASC(C$)
550 FOR I=1 TO 16
560 GET #2,A$: N$=N$+A$
570 NEXT
580 IF N$="" THEN 760: ELSE IF FT=0 THEN 750
590 OPEN 3,8,3,N$+LEFT$(F$(FT),1)+"R"
600 GET #3,A$,B$: CLOSE 3
610 SA=ASC(B$)*256+ASC(A$)
620 FOR I=1 TO 9: GET #2,A$: NEXT
630 GET #2,A$,B$: B=ASC(B$)*256+ASC(A$)
640 BB=BB+B
680 PRINT#4,N$ SPC(4)F$(FT) SPC(6)
690 PRINT#4,USING "###";B;
700 PRINT#4,SPC(5)"$" HEX$(SA)"/";
710 PRINT#4,USING "#####";SA;
720 PRINT#4,USING "{3SPACE}##";ET;
730 PRINT#4,USING "{5SPACE}##";ES;
740 PRINT#4,SPC(4)T1" "S1
750 Z=Z+32: GOTO 510
760 FOR I=1 TO 80: PRINT#4,"-";: NEXT : PRINT#4
770 PRINT#4,"BLOECKE:" SPC(5)"INSGESAMT = 664" S
    PC(5)"BELEGT = "BB;
780 PRINT#4,SPC(4)"FREI = "664-BB: PRINT#4
790 CLOSE 4: CLOSE 2: CLOSE 1
800 POKE 65286,PEEK(65286) OR 16
810 PRINT DS$: END
```

Listing 1. Das Programm zu »Directory-List«

Spitzen-Diskmonitor für C 16 und Plus/4

Mit »Disk Mon 16« wollen wir Ihnen einen durchaus professionellen Disketten-Monitor für den C16, den C116 und den Plus/4 anbieten. Es handelt sich hier sicherlich um das beste bisher veröffentlichte Programm.

Bei »Disk Mon 16« handelt es sich um ein professionelles Maschinenprogramm, das wir Ihnen als Basic-Lader anbieten (Listing 1). Angefangen bei Befehlen wie Block lesen und Block schreiben können Sie mit dem Disketten-Monitor auch alle Byte eines Blocks ändern, Befehle an das Diskettenlaufwerk senden, das Directory anzeigen und Zahlen in verschiedene Zahlensysteme umrechnen.

Wenn Sie das Programm (Listing 1) abgetippt und gespeichert haben, starten Sie es mit »RUN«. Die DATAs werden nun im Speicher des Computers als Maschinenprogramm abgelegt (das dauert etwa 96 Sekunden). Anschließend startet sich der Monitor automatisch.

Wenn alles stimmt, erscheint jetzt ein Titelbild auf dem Monitor, das das Hauptmenü enthält. Natürlich wollen Sie nicht jedesmal wieder das Basic-Programm laden und dann 96 Sekunden auf den Start des Disketten-Monitors warten. Sie können das eigentliche Maschinenprogramm nun auch direkt auf Diskette speichern und anstatt des Basic-Laders laden und starten, was folgendermaßen funktioniert:

Verlassen Sie mit <E> den Disketten-Monitor. Es erscheint »READY.«, und Sie befinden sich wieder im Basic. Anschließend tippen Sie

```
MONITOR
```

und drücken die <RETURN>-Taste. Haben Sie alles richtig gemacht, dann befinden Sie sich jetzt im Maschinensprache-Monitor Ihres Computers. Die Anzeige auf dem Bildschirm sieht etwa so aus:

```
MONITOR
   PC  SR AC XR YR SP
; FF00 00 00 FF 00 F2
```

Unterhalb dieser Zeilen blinkt der Cursor, und der Computer wartet auf Ihre Eingabe. Tippen Sie jetzt bitte

```
S"DISK MON 16",08,3835,3ED7
```

und drücken Sie wiederum die <RETURN>-Taste. Das Laufwerk läuft nun an und speichert den Disketten-Monitor auf Diskette. Das Ergebnis auf der Diskette ist ein Programm mit dem Namen »DISK MON 16«, das nur sieben Blöcke lang ist.

Speichern des Disk Mon 16

Wollen Sie das Programm auf der Datasette speichern, so müssen Sie die Angabe »,08,« hinter dem Filenamen im Maschinensprache-Monitor lediglich in »,01,« für die Datasette ändern.

Wollen Sie diesen »reinen« Disketten-Monitor nun wieder in den Computer laden und starten, so geben Sie

```
LOAD "DISK MON 16",8,1
```

beziehungsweise für die Datasette

```
LOAD "DISK MON 16",1,1
```

ein. Anschließend kommen Sie mit dem Befehl

```
SYS 14389
```

wieder in den »Disk Mon 16«.

Wir gehen einmal davon aus, daß der Disketten-Monitor gestartet ist. Dann erscheint das Hauptmenü mit allen zur Verfügung stehenden Befehlen.

Mit dem Befehl <M> ist es möglich, jederzeit wieder ins Hauptmenü zurückzukehren. Das ist zum Beispiel dann notwendig, wenn wir den Befehl für das Lesen eines Sektors nicht mehr wissen und noch einmal nachsehen wollen.

Mit dem Befehl <L> können Sie einen beliebigen Block von der Diskette in den Speicher des Computers lesen. Der Block steht dabei im Bereich von $3EFE bis $3FFD (16126 bis 16381). Die Syntax des <L>-Befehls lautet dabei

```
L Track Sektor
```

wobei für »Track« die Spurnummer (von 01 bis 35) und für Sektor die entsprechende Sektornummer (je nach Spur von 00 bis 20, 18, 17, 16) eingegeben werden kann.

Arbeit mit drei Zahlensystemen

Die Angabe der Track- und der Sektornummer kann dabei in einem der drei Zahlensysteme hexadezimal, dezimal oder binär eingegeben werden. Werden dezimale Werte verwendet, so erübrigt sich ein Kennzeichen. Bei hexadezimalen Werten müssen Sie ein <$> und bei Binärwerten ein <%> vor den betreffenden Wert setzen.

Wollen Sie einen Sektor wieder auf die Diskette schreiben, dann verwenden Sie den Befehl <S>. Seine Syntax ist gleich der des <L>-Befehls, wobei wiederum alle drei Zahlensysteme verwendet werden können, also zum Beispiel:

```
S $01 %00000000
```

Dieser Befehl schreibt einen Block bei Spur 1 Sektor 0 auf die Diskette. Die Tracknummer wurde dabei hexadezimal und die Sektornummer binär angegeben.

Haben Sie einen Block gelesen, dann wollen Sie sich dessen Inhalt natürlich auch gleich ansehen. Dazu dient der Befehl <A>. Wird kein Adreßbereich angegeben, so wird der gesamte Sektor aufgelistet. Die Angaben sind dabei alle hexadezimal.

Bremsen können Sie das Auflisten mit der Commodore-Taste, abbrechen können Sie mit <RUN/STOP>.

Wollen Sie nur eine bestimmte Anzahl von Bytes auf dem Bildschirm ausgeben, so geben Sie den Adreßbereich an, der angezeigt werden soll. Beide Angaben werden dabei hexadezimal entgegengenommen, wenn kein Kennzeichen angegeben wird und dürfen Werte zwischen $00 und $FF enthalten. Wollen Sie sich also nur die erste Hälfte eines Sektors ansehen, den Sie zuvor eingelesen haben, so tippen Sie

```
A 00 7F
```

oder aber auch

```
A %00000000 %01111111
```

Der nächste Befehl ist vor allem für Maschinensprache-Programmierer interessant. Es handelt sich um den Befehl <↑>, der das Springen in den Maschinensprache-Monitor gestattet. Hier kann der entsprechende Block dann zum Beispiel disassembliert oder assembliert werden. Die Rückkehr erfolgt entweder über Basic (siehe oben) oder mit »G 3835«.

Einen einmal eingelesenen Sektor können Sie natürlich auch verändern. Dazu lassen Sie sich den betreffenden Teil anzeigen (mit dem <A>-Befehl). Anschließend gehen Sie mit den Cursortasten auf das entsprechende Byte und

ändern es demgemäß ab. Die Eingabe wird in jeder Zeile mit <RETURN> abgeschlossen.

Wollen Sie eine Zeile direkt eingeben, dann schreiben Sie einen Doppelpunkt »:«, der dem Eingabe-Befehl entspricht, und geben anschließend eine Adresse und nachfolgend bis zu acht Byte ein. Wie Sie sicherlich bemerken, steht beim Auflisten eines Sektors an jedem Zeilenanfang schon automatisch der Doppelpunkt »:«, so daß ein einfaches Ändern der Bytes oder der Adresse genügt.

Natürlich ist es unter »Disk Mon 16« auch möglich, einen Befehl an die Diskettenlaufwerk zu senden. Dazu tippen Sie den Befehl <O> und nachfolgend die Zeichenkette für die Floppy ein. Wollen Sie also eine Diskette formatieren, dann tippen Sie zum Beispiel:

Befehl	Wirkung	Syntax
M	Anzeige des Hauptmenüs	M
L	Lesen eines Sektors	L Track Sektor
S	Schreiben eines Sektors	S Track Sektor
A	Anzeigen eines Blockinhalts	A Start-, Endadresse
↑	Sprung in den TEDMON	↑
:	Eingabe von Bytes	: Adresse Bytes....
O	Befehl an die Floppy senden	O Befehl
F	Fehlerkanal der Floppy auslesen	F
$	Directory anzeigen	$
Z	Umrechnen in andere Zahlensysteme	Z Wert
E	Rückkehr ins Basic	E

Tabelle 1. Komplette Übersicht über alle Befehle des Disketten-Monitors

```
ON:TESTDISKETTE,TD
```

Beim Arbeiten mit dem Laufwerk kommt es natürlich auch vor, daß zum Beispiel ein Fehler des Laufwerks durch das charakteristische Blinken angezeigt wird. Wollen Sie die Fehlermeldung im Klartext lesen, dann reicht die Eingabe von <F> und ein anschließendes <RETURN>, um den Fehlerkanal auszulesen und auf dem Bildschirm anzuzeigen.

Mit dem Befehl <$> können Sie sich das Directory der eingelegten Diskette anzeigen lassen.

Wollen Sie vielleicht auch einmal wissen, welcher Wert der hexadezimalen Zahl $6B entspricht? Dann kann Ihnen der Befehl <Z> weiterhelfen. Er erlaubt das Umrechnen einer Zahl aus einem der drei Zahlensysteme Dezimal, Hexadezimal und Binär in die jeweils anderen Systeme. Dazu ein Beispiel: Die Eingabe

```
Z $6B
```

ergibt den Ausdruck

```
HEX = $6B
DEZ = 107
BIN = %01101011
```

Ebensogut können Sie als Ausgangswert jedes der beiden anderen Zahlensysteme angeben. Binärzahlen erfordern dabei ein <%> vor dem Wert.

Die beiden letzten Befehle <M> und <E> wurden schon besprochen. Sie lassen das Hauptmenü (<M>) anzeigen oder ins Basic zurückkehren (<E>). Eine Aufstellung aller Befehle zeigt noch einmal Tabelle 1.

Für die Arbeit mit dem Disketten-Monitor gilt: generell ist nur die Angabe zweistelliger Hex-Werte bis $FF zulässig. Alle höheren Werte (dezimal 255 oder auch binär %11111111) führen zu Fehlfunktionen.

Vorsicht ist prinzipiell immer beim <S>-Befehl geboten. Ein einmal überschriebener Block kann nicht mehr zurückgeholt werden! (T. Pohl/ks)

```
10 REM
20 REM C16/C116 DISKMONITOR V3.5
30 REM
40 REM        L O A D E R
50 REM
60 :
65 POKE 55,14389 AND 255 : POKE.56,14389 / 256 :
   CLR
70 :
75 Z=1000: PS=0: AD=14389
80 :
90 PRINT "{RVSON,CLR,DOWN}BITTE 96 SEK. WARTEN!{
   RVOFF}"
95 :
100 FOR I=0 TO 7
105 : READ A$
110 : H=ASC(LEFT$(A$,1)): IF H>58 THEN H=H-55: G
    OTO 120
115 : H=H-48
120 : L=ASC(RIGHT$(A$,1)): IF L>58 THEN L=L-55:
    GOTO 130
125 : L=L-48
130 : P=L+(H*16)
135 : POKE AD+I,P
140 : PS=PS+P
145 NEXT I
150 READ B
155 IF B<>PS THEN PRINT "FEHLER IN ZEILE";Z : EN
    D
160 Z=Z+1: IF Z>1212 THEN SYS 14389
165 AD=AD+8
170 PS=0
175 GOTO 100
180 :
1000 DATA 20,7E,3B,A9,0D,20,D2,FF, 896
1001 DATA A9,5D,20,D2,FF,A9,00,85, 1061
1002 DATA 6A,20,58,39,C9,5D,F0,F9, 1066
1003 DATA C9,20,F0,F5,A2,0C,DD,22, 1147
1004 DATA 3B,D0,11,86,69,BD,38,3B, 827
1005 DATA 8D,68,38,BD,2D,3B,8D,67, 838
1006 DATA 38,4C,38,38,CA,10,E7,A9, 862
1007 DATA 03,85,CA,A2,02,BD,7E,38, 873
1008 DATA 20,D2,FF,CA,10,F7,4C,38, 1094
1009 DATA 38,83,3F,82,85,96,A9,20, 864
1010 DATA 20,D2,FF,B9,FE,3E,20,13, 1049
1011 DATA 39,C8,D0,02,E6,6A,C6,96, 1151
1012 DATA D0,EC,60,4C,38,38,4C,6C, 912
1013 DATA 38,20,F5,3C,90,12,85,6C, 796
1014 DATA A5,DA,C9,0D,F0,F0,20,F5, 1354
1015 DATA 3C,90,05,85,6D,4C,BD,38, 772
1016 DATA A0,00,84,6C,A0,FF,84,6D, 1056
1017 DATA A4,6C,A5,6A,D0,D5,C4,6D, 1269
1018 DATA B0,D1,20,4B,3B,A9,0D,20, 765
1019 DATA D2,FF,A9,5D,20,D2,FF,A9, 1393
1020 DATA 3A,20,D2,FF,98,20,13,39, 815
1021 DATA A9,20,20,D2,FF,A9,08,20, 907
1022 DATA 81,38,20,EF,3A,4C,BF,38, 837
1023 DATA 20,31,39,B0,03,4C,38,38, 505
1024 DATA A8,A9,08,85,96,20,58,39, 805
1025 DATA 20,58,39,20,31,39,90,03, 462
1026 DATA 99,FE,3E,C8,C6,96,D0,F0, 1465
1027 DATA 20,EF,3A,4C,38,38,48,4A, 663
1028 DATA 4A,4A,4A,20,27,39,20,D2, 592
1029 DATA FF,68,29,0F,20,27,39,4C, 619
1030 DATA D2,FF,18,69,F6,90,02,69, 1091
1031 DATA 06,69,3A,60,A9,00,85,6B, 674
1032 DATA 20,58,39,C9,20,D0,09,20, 659
1033 DATA 58,39,C9,20,D0,0E,18,60, 720
1034 DATA 20,DA,3D,0A,0A,0A,0A,85, 484
1035 DATA 6B,20,58,39,20,DA,3D,05, 600
1036 DATA 6B,38,60,20,CF,FF,C9,0D, 967
1037 DATA D0,F8,4C,38,38,20,CF,FF, 1138
1038 DATA C9,0D,D0,03,4C,6C,38,C9, 866
1039 DATA 20,F0,F2,A2,00,9D,D6,3E, 1109
1040 DATA E8,E0,28,F0,07,20,CF,FF, 1237
1041 DATA C9,0D,D0,F1,A9,0D,9D,D6, 1216
1042 DATA 3E,A2,00,BD,CA,39,20,D2, 914
```

Listing 1. Der »Disk Mon 16« als Basic-Programm

```
1043 DATA FF,E8,E0,1B,D0,F5,A9,1B, 1384
1044 DATA 85,CA,20,CF,FF,C9,4A,F0, 1344
1045 DATA 03,4C,38,38,A9,0D,20,D2, 615
1046 DATA FF,A9,08,85,AE,20,B1,FF, 1203
1047 DATA A9,6F,85,AD,20,93,FF,A2, 1182
1048 DATA 00,BD,D6,3E,20,A8,FF,E8, 1152
1049 DATA BD,D6,3E,C9,0D,D0,F5,20, 1164
1050 DATA AE,FF,4C,55,3B,0D,82,11, 809
1051 DATA 53,49,4E,44,20,53,49,45, 559
1052 DATA 20,53,49,43,48,45,52,20, 510
1053 DATA 3F,20,28,4A,2F,4E,29,83, 506
1054 DATA 20,F5,3C,B0,03,4C,6C,38, 756
1055 DATA 8D,91,3A,A5,DA,C9,0D,F0, 1181
1056 DATA 05,20,F5,3C,B0,03,4C,6C, 705
1057 DATA 38,8D,94,3A,20,B3,3A,A5, 837
1058 DATA 69,C9,01,F0,25,A9,31,20, 834
1059 DATA 57,3A,A2,0D,20,C6,FF,A2, 967
1060 DATA 00,20,CF,FF,9D,FE,3E,E8, 1199
1061 DATA D0,F7,20,CC,FF,A9,0D,20, 1160
1062 DATA C3,FF,A9,0F,20,C3,FF,4C, 1192
1063 DATA 38,38,20,96,3A,A2,0D,20, 559
1064 DATA C9,FF,A2,00,BD,FE,3E,20, 1155
1065 DATA D2,FF,E8,D0,F7,20,CC,FF, 1643
1066 DATA A9,32,20,57,3A,A9,0D,20, 610
1067 DATA C3,FF,A9,0F,20,C3,FF,4C, 1192
1068 DATA 38,38,8D,8A,3A,A2,0F,AD, 799
1069 DATA 91,3A,20,E2,3A,8E,91,3A, 864
1070 DATA 8D,92,3A,AD,94,3A,20,E2, 982
1071 DATA 3A,8E,94,3A,8D,95,3A,A2, 916
1072 DATA 0F,20,C9,FF,A2,00,BD,89, 991
1073 DATA 3A,20,D2,FF,E8,E0,0D,D0, 1232
1074 DATA F5,4C,CC,FF,55,31,3A,31, 1021
1075 DATA 33,20,30,20,00,00,20,00, 195
1076 DATA 00,A2,0F,20,C9,FF,A2,00, 827
1077 DATA BD,AB,3A,20,D2,FF,E8,E0, 1371
1078 DATA 08,D0,F5,4C,CC,FF,42,2D, 1107
1079 DATA 50,20,31,33,20,30,A9,0F, 476
1080 DATA A8,A2,08,20,BA,FF,A9,00, 980
1081 DATA 20,BD,FF,20,C0,FF,A9,0D, 1137
1082 DATA A8,A2,08,20,BA,FF,A9,01, 981
1083 DATA A2,D7,A0,3A,20,BD,FF,4C, 1147
1084 DATA C0,FF,23,A9,0D,20,C3,FF, 1146
1085 DATA A9,0F,4C,C3,FF,A2,30,38, 976
1086 DATA E9,0A,90,03,E8,B0,F9,69, 1152
1087 DATA 3A,60,98,38,E9,08,A8,A9, 940
1088 DATA 20,20,D2,FF,A9,12,20,D2, 958
1089 DATA FF,A2,08,B9,FE,3E,29,7F, 1094
1090 DATA C9,20,B0,04,A9,2E,D0,03, 839
1091 DATA B9,FE,3E,20,D2,FF,C8,CA, 1400
1092 DATA D0,E9,A9,92,4C,D2,FF,20, 1329
1093 DATA BC,C8,4C,38,38,4C,53,41, 800
1094 DATA 5E,3A,4F,46,24,5A,4D,45, 573
1095 DATA E5,E5,9E,43,ED,62,55,1C, 1131
1096 DATA 0D,35,0A,39,39,38,3B,38, 361
1097 DATA 39,3B,3B,3E,38,80,A9,93, 737
1098 DATA 20,D2,FF,4C,52,FF,A6,91, 1221
1099 DATA E0,7F,D0,03,4C,38,38,60, 846
1100 DATA A9,00,85,90,A9,0D,20,D2, 870
1101 DATA FF,A9,08,85,AE,20,B4,FF, 1206
1102 DATA A9,6F,85,AD,20,96,FF,20, 1055
1103 DATA A5,FF,24,90,70,05,20,D2, 959
1104 DATA FF,D0,F4,20,AB,FF,4C,38, 1297
1105 DATA 38,A0,00,A9,A8,A2,3B,85, 707
1106 DATA D8,86,D9,78,8C,15,FF,8C, 1243
1107 DATA 19,FF,58,B1,D8,20,D2,FF, 1258
1108 DATA E6,D8,D0,02,E6,D9,A5,D8, 1484
1109 DATA C9,F4,D0,EF,A5,D9,C9,3C, 1535
1110 DATA D0,E9,60,99,93,11,20,20, 918
1111 DATA 43,31,36,2F,43,31,31,36, 436
1112 DATA 20,44,49,53,4B,2D,4D,4F, 532
1113 DATA 4E,49,54,4F,52,20,56,33, 565
1114 DATA 2E,35,0D,0D,20,20,20,20, 253
1115 DATA 20,20,28,57,29,20,34,2F, 363
1116 DATA 38,36,20,42,59,20,54,2E, 459
1117 DATA 50,4F,48,4C,0D,0D,0D,2D, 391
1118 DATA 2D,2D,2D,2D,2D,2D,2D,2D, 360
1119 DATA 2D,2D,2D,4D,45,4E,55,45, 513
1120 DATA 2D,2D,2D,2D,2D,2D,2D,2D, 360
1121 DATA 2D,2D,2D,2D,2D,0D,0D,5D, 344
1122 DATA 4C,20,3D,20,42,4C,4F,43, 489
1123 DATA 4B,20,4C,45,53,45,4E,0D, 495
1124 DATA 5D,53,20,3D,20,42,4C,4F, 522
1125 DATA 43,4B,20,53,43,48,52,45, 547
1126 DATA 49,42,45,4E,0D,5D,41,20, 489
1127 DATA 3D,20,42,4C,4F,43,4B,20, 488
1128 DATA 41,4E,5A,45,49,47,45,4E, 593
```

```
1129 DATA 0D,5D,5E,20,3D,20,56,45, 480
1130 DATA 52,5A,57,45,49,47,54,20, 588
1131 DATA 49,4E,20,43,42,4D,20,4D, 502
1132 DATA 4F,4E,49,54,4F,52,0D,5D, 581
1133 DATA 3A,20,3D,20,42,59,54,45, 491
1134 DATA 53,20,41,45,4E,44,45,52, 546
1135 DATA 4E,0D,5D,4F,20,3D,20,46, 458
1136 DATA 4C,4F,50,50,59,2D,42,45, 584
1137 DATA 46,45,48,4C,20,53,45,4E, 549
1138 DATA 44,45,4E,0D,5D,46,20,3D, 484
1139 DATA 20,46,45,48,4C,45,52,20, 502
1140 DATA 41,4E,5A,45,49,47,45,4E, 593
1141 DATA 0D,5D,24,20,3D,20,44,49, 408
1142 DATA 52,45,43,54,4F,52,59,20, 584
1143 DATA 41,4E,5A,45,49,47,45,4E, 593
1144 DATA 0D,5D,5A,20,3D,20,5A,41, 476
1145 DATA 48,4C,45,4E,53,59,53,54, 634
1146 DATA 45,4D,45,0D,5D,4D,20,3D, 491
1147 DATA 20,4D,45,4E,55,45,20,41, 507
1148 DATA 4E,5A,45,49,47,45,4E,0D, 541
1149 DATA 5D,45,20,3D,20,5A,55,52, 544
1150 DATA 55,45,43,4B,20,49,4E,53, 562
1151 DATA 20,42,41,53,49,43,0D,00, 399
1152 DATA A9,00,85,6B,20,CF,FF,C9, 1104
1153 DATA 20,F0,F5,C9,0D,D0,02,18, 965
1154 DATA 60,C9,24,F0,39,C9,25,F0, 1108
1155 DATA 53,20,B6,3D,E0,01,F0,26, 861
1156 DATA E0,02,F0,14,AD,85,3D,20, 885
1157 DATA E6,3D,AD,86,3D,20,F4,3D, 996
1158 DATA AD,87,3D,20,02,3E,38,60, 617
1159 DATA AD,85,3D,20,F4,3D,AD,86, 1011
1160 DATA 3D,20,02,3E,38,60,AD,85, 615
1161 DATA 3D,20,02,3E,38,60,20,B3, 520
1162 DATA 3D,AD,85,3D,20,DA,3D,0A, 749
1163 DATA 0A,0A,0A,8D,82,3D,AD,86, 669
1164 DATA 3D,20,DA,3D,0D,82,3D,8D, 717
1165 DATA 82,3D,38,60,20,B3,3D,8E, 757
1166 DATA 83,3D,A2,00,BD,85,3D,20, 769
1167 DATA DA,3D,29,01,0D,82,3D,8D, 666
1168 DATA 82,3D,0E,82,3D,E8,EC,83, 995
1169 DATA 3D,D0,E9,38,60,00,00,00, 654
1170 DATA 00,00,00,00,00,00,00,00, 0
1171 DATA 00,A2,64,2C,A2,0A,8E,A7, 787
1172 DATA 3D,8D,8D,3D,A9,00,8D,84, 846
1173 DATA 3D,A2,08,4E,8D,3D,90,03, 658
1174 DATA 18,69,00,6A,6E,84,3D,CA, 740
1175 DATA D0,F1,AD,84,3D,60,20,CF, 1150
1176 DATA FF,A2,00,9D,85,3D,E8,E0, 1224
1177 DATA 08,F0,12,20,CF,FF,C9,20, 993
1178 DATA D0,07,C9,47,B0,07,4C,D2, 956
1179 DATA 3D,C9,0D,D0,E6,85,DA,A9, 1233
1180 DATA 00,8D,82,3D,60,C9,41,90, 838
1181 DATA 04,38,E9,37,60,38,E9,30, 781
1182 DATA 60,20,DA,3D,20,8E,3D,18, 666
1183 DATA 6D,82,3D,8D,82,3D,60,20, 760
1184 DATA DA,3D,20,91,3D,18,6D,82, 780
1185 DATA 3D,8D,82,3D,60,20,DA,3D, 800
1186 DATA 18,6D,82,3D,8D,82,3D,60, 752
1187 DATA 20,F5,3C,B0,03,4C,6C,38, 756
1188 DATA 8D,C5,3E,A2,00,BD,AD,3E, 986
1189 DATA 20,D2,FF,E8,E0,09,D0,F5, 1415
1190 DATA AD,C5,3E,29,F0,4A,4A,4A, 935
1191 DATA 4A,AA,BD,C6,3E,20,D2,FF, 1190
1192 DATA AD,C5,3E,29,0F,AA,BD,C6, 1045
1193 DATA 3E,20,D2,FF,A2,00,BD,B6, 1092
1194 DATA 3E,20,D2,FF,E8,E0,07,D0, 1230
1195 DATA F5,A9,30,85,61,85,62,85, 1056
1196 DATA 63,A2,00,AC,C5,3E,C0,00, 884
1197 DATA F0,06,20,99,3E,88,D0,FA, 1087
1198 DATA A2,02,B5,61,20,D2,FF,CA, 1141
1199 DATA 10,F8,A2,00,BD,BD,3E,20, 898
1200 DATA D2,FF,E8,E0,08,D0,F5,A2, 1544
1201 DATA 08,AD,C5,3E,0A,48,90,03, 669
1202 DATA A9,31,2C,A9,30,20,D2,FF, 976
1203 DATA 68,CA,D0,F0,A9,0D,20,D2, 1178
1204 DATA FF,4C,38,38,F6,61,B5,61, 1064
1205 DATA C9,3A,F0,01,60,A9,30,95, 962
1206 DATA 61,E8,20,99,3E,A2,00,60, 834
1207 DATA 0D,11,48,45,58,20,3D,20, 384
1208 DATA 24,0D,44,45,5A,20,3D,20, 401
1209 DATA 0D,42,49,4E,20,3D,20,25, 392
1210 DATA 00,30,31,32,33,34,35,36, 357
1211 DATA 37,38,39,41,42,43,44,45, 503
1212 DATA 46,00,00,FF,FF,00,00,FF, 835
```

Listing 1. Basic-Lader zum bequemeren Eingeben des »Disk Mon 16«

Super Copy für C 16 und Plus/4

Nun gibt es auch für alle C 16- und Plus/4-Besitzer ein File Copy. Mit diesem komfortablen Programm ist es ein Kinderspiel, sich von wichtigen Daten eine Sicherheitskopie zu machen.

Das Programm »Super Copy« gibt es in einer neuen Version jetzt auch für den C 16 (allerdings nur für die 64-KByte-Version) und den Plus/4, wobei sowohl das Floppy-Laufwerk 1541 als auch die neue 1551 angeschlossen werden können. Die 1551 hat dabei natürlich den Vorteil der höheren Verarbeitungsgeschwindigkeit.

Super Copy finden Sie in Listing 1 als Ausdruck mit dem eingebauten Monitor. Da uns für den C 16 und den Plus/4 leider noch kein MSE zur Verfügung steht, müssen wir an Ihre Vorsicht beim Abtippen des Programms appellieren.

Haben Sie noch nicht mit dem Monitor gearbeitet, dann möchten wir Sie an dieser Stelle auf unseren Kurs über den eingebauten Maschinensprache-Monitor des C 16 und des Plus/4 verweisen. Dieser Artikel befindet sich auch in dieser Ausgabe und erläutert Ihnen die Bedienung des sehr hilfreichen Werkzeugs. Sie werden erkennen, daß Sie auch als Basic-Programmierer die Vorteile des Monitors nutzen können und daß dessen Bedienung einfacher ist, als man denkt.

Haben Sie die Eingabe beendet, dann speichern Sie das Programm bitte vor dem Start. Das geschieht im Maschinensprache-Monitor auf die folgende Weise:

```
S"SUPER COPY",08,1001,1B00
```

Nach Eingabe der Zeile drücken Sie <RETURN>, und Super Copy wird auf die Diskette gespeichert. Wollen Sie ein Speichern auf der Datasette, so setzen Sie hinter dem Filenamen statt »,08,« den Wert »,01,« ein.

Drücken Sie jetzt am besten den Reset-Knopf an Ihrem Computer und laden Sie Super Copy von neuem.

Super Copy steht am Start des Basic-Speichers und enthält eine einzige Basic-Zeile. Der Start erfolgt mit <RUN>, worauf Sie sich im Hauptmenü wiederfinden.

Jetzt haben Sie sechs Möglichkeiten zur Auswahl. Sie können sich unter <1> das Directory der eingelegten Diskette anzeigen lassen. Punkt <3> dient dem Formatieren einer Diskette, worauf Sie den Diskettennamen und die zweistellige ID angeben müssen. Bei der Wahl von <4> können Sie mehrere Files von der Diskette löschen. Sie werden dabei aufgefordert, die entsprechende Diskette einzulegen und sich die einzelnen Files durch Angabe von <J> für Ja und <N> für Nein auszusuchen. Bei der Angabe von Ja wird das entsprechende File dann anschließend gelöscht.

Bei der Auswahl von <5> im Hauptmenü können Sie eine Diskette validieren. Genaueres zu diesem Befehl steht im Handbuch zu Ihrem Diskettenlaufwerk.

Entscheidend ist der Menüpunkt <2>. Hier werden Sie wie schon bei Punkt <4> aufgefordert, eine Diskette einzulegen und anschließend die Files durch Angabe von »J« und »N« für das Kopieren herauszusuchen. Haben Sie diesen Vorgang für das gesamte Directory durchgeführt, so können Sie sich noch einmal das Directory anzeigen lassen, eine Diskette formatieren oder validieren.

Mit <SPACE> können Sie den Kopiervorgang sofort beginnen, wobei die Files von der Quelldiskette nun in den Speicher des Computers gelesen werden.

Anschließend werden Sie gefragt, ob der Kopiervorgang fortlaufend oder einzeln erfolgen soll. Beim fortlaufenden Betrieb werden alle Files auf eine Zieldiskette kopiert. Ansonsten können Sie auch verschiedene Zieldisketten verwenden.

Jetzt erfolgt nach einer Abfrage die Aufforderung, die Zieldiskette einzulegen.

Wollen Sie mehr Files kopieren als auf einmal in den Computer passen, so können bis zu drei Diskettenwechsel notwendig werden. Der Computer fordert Sie in diesem Fall automatisch dazu auf. Alle Wechsel sind dabei jeweils durch einen Tastendruck zu bestätigen.

(H.J. Knacke/ks)

```
>1001 0C 10 0A 00 9E 20 34 31 :
>1009 31 32 00 00 00 FF 02 A9 :
>1011 3D 8D 15 FF 8D 19 FF A9 :
>1019 73 8D 3B 05 A9 8B A0 17 :
>1021 20 0B 17 20 E7 16 C9 31 :
>1029 90 F9 C9 37 B0 F5 20 D2 :
>1031 FF C9 31 D0 06 20 61 10 :
>1039 4C 10 10 C9 32 D0 03 4C :
>1041 C2 10 C9 33 D0 06 20 E6 :
>1049 10 4C 10 10 C9 34 D0 03 :
>1051 4C 4E 11 C9 35 D0 06 20 :
>1059 B9 11 4C 10 10 4C 7E 86 :
>1061 20 67 C5 20 BC C8 20 E4 :
>1069 FF F0 FB 60 EA EA EA EA :
>1071 EA EA EA EA EA EA EA EA :
>1079 EA EA EA EA EA EA EA 18 :
>1081 7D A0 1B B0 05 F0 03 4C :
>1089 D0 14 60 EA EA EA EA EA :
>1091 EA EA EA EA EA EA EA EA :
>1099 EA EA EA EA EA EA EA EA :
>10A1 EA EA EA EA EA EA EA EA :
>10A9 EA EA EA EA EA EA EA EA :
>10B1 EA EA EA EA EA EA EA EA :
>10B9 EA EA EA EA EA EA EA EA :
>10C1 EA 20 D1 11 A9 6F A0 1A :
>10C9 20 0B 17 A9 FF 8D 03 04 :
>10D1 20 75 13 20 DE 13 20 C3 :
>10D9 13 20 75 13 20 EA 14 20 :
>10E1 E3 13 4C DD 10 A9 A8 A0 :
>10E9 19 20 0B 17 A2 00 20 CF :
>10F1 FF 9D 40 03 E8 C9 0D D0 :
>10F9 F5 A9 00 CA 9D 40 03 A9 :
>1101 4E 8D 3D 03 A9 30 8D 3E :
>1109 03 A9 3A 8D 3F 03 A9 0F :
>1111 A2 08 A8 20 BA FF A2 03 :
>1119 BD 3D 03 F0 03 E8 D0 F8 :
>1121 8A A2 3D A0 03 20 BD FF :
>1129 20 C0 FF 20 A3 16 08 A9 :
>1131 0F 20 C3 FF 28 B0 01 60 :
>1139 A9 C8 A0 19 20 0B 17 20 :
>1141 F5 16 C9 4A F0 B9 C9 4E :
>1149 D0 F5 4C 10 10 20 67 C5 :
>1151 A9 20 A0 1A 20 0B 17 A9 :
>1159 80 8D 04 04 20 E0 11 A9 :
>1161 08 20 C3 FF AD 01 04 D0 :
>1169 03 4C 10 10 20 44 13 A9 :
>1171 87 A0 1A 20 0B 17 20 DE :
>1179 13 A9 53 8D 3D 03 A9 30 :
>1181 8D 3E 03 A9 3A 8D 3F 03 :
>1189 8A 18 69 03 A2 3D A0 03 :
>1191 20 BD FF A9 0F A2 08 A8 :
```

Listing 1. »Super Copy«

```
>1199 20 BA FF 20 C0 FF A9 0F
>11A1 20 C3 FF AE 00 04 E8 EC
>11A9 01 04 B0 09 EE 00 04 20
>11B1 12 14 4C 7A 11 4C 10 10
>11B9 A9 56 8D EF 1A A9 1B A0
>11C1 19 20 0B 17 20 4B 16 A9
>11C9 49 8D EF 1A 20 88 16 60
>11D1 20 67 C5 A9 00 8D 04 04
>11D9 A9 2E A0 19 20 0B 17 20
>11E1 F5 16 20 F0 16 20 F0 16
>11E9 20 4B 16 20 88 16 90 05
>11F1 68 68 4C 10 10 A9 08 AA
>11F9 A0 00 20 BA FF A9 01 A2
>1201 ED A0 1A 20 BD FF 20 C0
>1209 FF A9 08 20 B4 FF A9 00
>1211 20 96 FF A0 04 20 A5 FF
>1219 88 D0 FA 20 A5 FF 85 D8
>1221 20 A5 FF A6 D8 20 5F A4
>1229 20 ED 16 20 A5 FF F0 06
>1231 20 D2 FF 18 90 F5 20 F0
>1239 16 20 F0 16 20 A5 FF 20
>1241 A5 FF A0 00 8C 01 04 20
>1249 A5 FF 8D 00 04 20 A5 FF
>1251 8D 02 04 AE 00 04 20 5F
>1259 A4 20 ED 16 A0 00 20 A5
>1261 FF 20 D2 FF 99 40 03 F0
>1269 03 C8 D0 F2 20 A5 FF 20
>1271 A5 FF A5 90 F0 09 2C 04
>1279 04 10 01 60 4C 32 13 AD
>1281 02 04 D0 07 AD 00 04 C9
>1289 DF 90 0A A9 46 A0 18 20
>1291 0B 17 4C 2C 13 AD 01 04
>1299 C9 20 90 0A A9 56 A0 18
>12A1 20 0B 17 18 90 D6 A9 00
>12A9 85 08 A9 20 85 CA A9 6E
>12B1 A0 18 20 0B 17 20 F5 16
>12B9 C9 4E F0 68 C9 4A D0 F5
>12C1 A9 7D A0 18 20 0B 17 AD
>12C9 01 04 20 E9 15 EA EA A2
>12D1 00 E8 BD 40 03 C9 22 D0
>12D9 F8 8E 02 04 E8 BD 40 03
>12E1 C9 22 F0 07 91 41 EA E8
>12E9 C8 D0 F2 8A AC 01 04 18
>12F1 ED 02 04 99 00 1B AD 00
>12F9 04 99 A0 1B BD 40 03 D0
>1301 0F 2C 04 04 30 15 A9 87
>1309 A0 18 20 0B 17 4C 2C 13
>1311 C9 53 F0 07 C9 50 F0 03
>1319 E8 D0 E1 99 20 1B EE 01
>1321 04 18 90 07 A9 AA A0 18
>1329 20 0B 17 20 F0 16 4C 48
>1331 12 20 AB FF A9 08 20 C3
>1339 FF AD 01 04 D0 05 68 68
>1341 4C 10 10 A2 00 8E 06 04
>1349 8E 05 04 20 F0 16 20 F0
>1351 16 A9 12 20 D2 FF 18 BD
>1359 A0 1B 6D 05 04 8D 05 04
>1361 A9 00 6D 06 04 8D 06 04
>1369 E8 EC 01 04 90 E9 AE 05
>1371 04 4C 5F A4 A9 EA A0 18
>1379 20 0B 17 2C 03 04 30 07
>1381 A9 D7 A0 1A 20 0B 17 A9
>1389 A0 A0 1A 20 0B 17 20 F5
>1391 16 C9 20 D0 01 60 C9 32
>1399 D0 06 20 B9 11 4C 75 13
>13A1 C9 31 D0 06 20 61 10 4C
>13A9 75 13 C9 33 D0 06 20 E6
>13B1 10 4C 75 13 C9 34 D0 D6
>13B9 2C 03 04 30 D1 68 68 4C
>13C1 F4 14 A9 B4 A0 18 20 0B
>13C9 17 20 F5 16 C9 31 F0 07
>13D1 C9 32 D0 F5 A9 00 2C A9
>13D9 FF 8D 03 04 60 A2 00 BE
>13E1 00 04 A9 1E 9D 40 1B A9
>13E9 00 8D 02 04 2C 04 04 10
>13F1 0A A9 3E A0 1A 20 0B 17
>13F9 18 90 0A 8A F0 1D A9 2E
>1401 A0 19 20 0B 17 20 F5 16
>1409 2C 04 04 10 0E C9 20 D0
>1411 F4 A9 60 A0 1A 20 0B 17
>1419 18 90 07 A9 4E A0 19 20
>1421 0B 17 AC 00 04 BE 00 1B
>1429 AD 00 04 20 E9 15 B1 41
>1431 20 D2 FF C8 CA D0 F7 20
>1439 1C 17 AE 00 04 BD 00 1B
>1441 85 D8 AD 00 04 20 E9 15
>1449 A2 00 B1 41 9D 40 03 C8
>1451 E8 C6 D8 D0 F5 2C 04 04
>1459 10 01 60 A0 00 B9 F1 1A
>1461 9D 40 03 C8 E8 C0 04 90
>1469 F4 AC 00 04 B9 20 1B 9D
>1471 3D 03 8A A2 40 A0 03 20
>1479 BD FF A9 02 A2 08 A8 20
>1481 BA FF 20 C0 FF AE 00 04
>1489 BD 40 1B A0 00 84 D8 85
>1491 D9 A2 02 20 C6 FF 20 A5
>1499 FF 20 09 16 A6 90 F0 F6
>14A1 20 88 16 08 20 CC FF A9
>14A9 02 20 C3 FF 28 90 05 20
>14B1 D1 16 B0 2C AE 00 04 38
>14B9 A5 D8 E9 01 9D 80 1B A5
>14C1 D9 E9 00 9D 60 1B E8 EC
>14C9 01 04 B0 1C 4C 80 10 EA
>14D1 EA EA EE 00 04 EE 02 04
>14D9 A5 D9 18 69 01 9D 40 1B
>14E1 A9 0F 20 C3 FF 4C 1C 14
>14E9 60 AD 00 04 38 ED 02 04
>14F1 8D 00 04 A9 5A A0 19 20
>14F9 0B 17 20 F5 16 20 4B 16
>1501 A9 79 A0 19 20 0B 17 AC
>1509 00 04 BE 00 1B AD 00 04
>1511 20 E9 15 EA B1 41 EA 20
>1519 D2 FF C8 CA D0 F6 20 1C
>1521 17 AE 00 04 BD 00 1B 85
>1529 D8 AD 00 04 20 E9 15 A2
>1531 00 B1 41 9D 40 03 E8 C8
>1539 C6 D8 D0 F5 A0 00 B9 F6
>1541 1A 9D 40 03 C8 E8 C0 04
>1549 90 F4 AC 00 04 B9 20 1B
>1551 9D 3D 03 8A A2 40 A0 03
>1559 20 BD FF A9 02 A2 08 A8
>1561 20 BA FF 20 C0 FF AE 00
>1569 04 A0 00 BD 40 1B 84 D8
>1571 85 D9 BD 80 1B 85 DA BD
>1579 60 1B 85 DB A2 02 20 C9
>1581 FF 20 1A 16 20 A8 FF A5
>1589 DA C5 D8 A5 DB E5 D9 B0
>1591 F0 20 CC FF A9 02 20 C3
>1599 FF 20 88 16 90 05 20 D1
>15A1 16 B0 10 2C 03 04 30 03
>15A9 4C D5 15 CE 02 04 30 06
>15B1 EE 00 04 4C FE 14 EE 00
>15B9 04 AE 00 04 EC 01 04 B0
>15C1 04 20 F0 16 60 A9 85 A0
>15C9 19 20 0B 17 20 F5 16 68
>15D1 68 4C 10 10 20 75 13 CE
>15D9 02 04 10 03 4C B7 15 EE
>15E1 00 04 20 22 16 4C F4 14
>15E9 A0 00 0A 0A 84 42 0A 26
>15F1 42 0A 26 42 85 41 A5 42
>15F9 18 69 1C 85 42 A0 00 60
>1601 78 8C 3F FF A0 00 EA 60
>1609 20 01 16 91 D8 E6 D8 D0
>1611 02 E6 D9 8C 3E FF EA 58
>1619 60 20 01 16 B1 D8 4C 0E
>1621 16 A9 0C A0 1A 20 0B 17
>1629 AD 00 04 0A 0A 0A 0A 08
>1631 AE 00 04 BC 00 1B AA BD
>1639 00 1C 28 08 90 03 BD 00
>1641 1D 20 D2 FF E8 88 D0 EF
>1649 28 60 A9 0F A2 08 A8 20
>1651 BA FF A9 01 A2 EF A0 1A
```

```
>1659 20 BD FF 20 C0 FF A9 0F :
>1661 4C C3 FF 20 A5 FF 29 0F :
>1669 0A 0A 0A 0A 85 57 20 A5 :
>1671 FF 29 0F 05 57 60 48 4A :
>1679 4A 4A 4A 20 80 16 68 29 :
>1681 0F 18 69 30 4C D2 FF A9 :
>1689 0F A2 08 A8 20 BA FF A9 :
>1691 00 20 BD FF 20 C0 FF 20 :
>1699 A3 16 08 A9 0F 20 C3 FF :
>16A1 28 60 A2 0F 20 C6 FF 20 :
>16A9 64 16 C9 01 08 90 0B 48 :
>16B1 20 F0 16 20 F0 16 68 20 :
>16B9 77 16 20 A5 FF C9 0D F0 :
>16C1 0A 28 08 90 F5 20 D2 FF :
>16C9 18 90 EF 20 CC FF 28 60 :
>16D1 A9 DF A0 19 20 0B 17 20 :
>16D9 F5 16 C9 31 D0 02 18 60 :
>16E1 C9 32 D0 F3 38 60 20 E4 :
>16E9 FF F0 FB 60 A9 20 2C A9 :
>16F1 0D 4C D2 FF 20 E7 16 C9 :
>16F9 5C F0 01 60 A9 0F 20 C3 :
>1701 FF 20 E7 FF A2 F7 9A 4C :
>1709 10 10 85 57 84 58 A0 00 :
>1711 B1 57 F0 06 20 D2 FF C8 :
>1719 D0 F6 60 A2 00 8E 40 03 :
>1721 8E 41 03 AE 00 04 BD A0 :
>1729 1B A2 08 48 68 0A 48 F8 :
>1731 AD 41 03 6D 41 03 8D 41 :
>1739 03 AD 40 03 6D 40 03 8D :
>1741 40 03 D8 CA D0 E6 68 AD :
>1749 40 03 09 30 8D 40 03 AD :
>1751 41 03 29 0F 09 30 8D 42 :
>1759 03 AD 41 03 4A 4A 4A 4A :
>1761 09 30 8D 41 03 A9 1D 85 :
>1769 CA A2 00 BD 40 03 C9 30 :
>1771 D0 0A A9 20 9D 40 03 E8 :
>1779 E0 03 90 EF A2 00 BD 40 :
>1781 03 20 D2 FF E8 E0 03 90 :
>1789 F5 60 93 0D 0D 20 20 20 :
>1791 20 20 20 20 2A 2A 2A 2A :
>1799 20 53 55 50 45 52 20 43 :
>17A1 4F 50 59 20 31 35 34 31 :
>17A9 20 2A 2A 2A 2A 0D 20 20 :
>17B1 20 20 20 20 20 2A 2A 2A :
>17B9 2A 20 43 31 36 2F 31 31 :
>17C1 36 20 36 34 4B 20 52 41 :
>17C9 4D 20 2A 2A 2A 2A 0D 0D :
>17D1 0D 0D 20 20 31 2E 20 44 :
>17D9 49 52 45 43 54 4F 52 59 :
>17E1 0D 20 20 32 2E 20 4B 4F :
>17E9 50 49 45 52 45 4E 0D 20 :
>17F1 20 33 2E 20 46 4F 52 4D :
>17F9 41 54 49 45 52 45 4E 0D :
>1801 20 20 34 2E 20 53 43 52 :
>1809 41 54 43 48 0D 20 20 35 :
>1811 2E 20 56 41 4C 49 44 49 :
>1819 45 52 45 4E 0D 20 20 36 :
>1821 2E 20 45 4E 44 45 0D 0D :
>1829 0D 0D 20 20 12 20 42 49 :
>1831 54 54 45 20 57 41 45 48 :
>1839 4C 45 4E 20 53 49 45 20 :
>1841 3A 20 92 20 00 0D 0D 20 :
>1849 46 49 4C 45 20 5A 55 20 :
>1851 4C 41 4E 47 00 0D 0D 20 :
>1859 12 20 4B 4F 50 49 45 52 :
>1861 4C 49 53 54 45 20 56 4F :
>1869 4C 4C 20 92 00 4A 41 2F :
>1871 4E 45 49 4E 9D 9D 9D 9D :
>1879 9D 9D 9D 00 12 20 4A 41 :
>1881 20 92 20 20 20 00 0D 20 :
>1889 20 12 20 46 41 4C 53 43 :
>1891 48 45 52 20 46 49 4C 45 :
>1899 54 59 50 20 92 20 20 20 :
>18A1 20 20 12 5E 5E 5E 92 0D :
>18A9 00 12 4E 45 49 4E 92 20 :
>18B1 20 20 00 0D 0D 0D 0D 20 :
>18B9 4B 4F 50 49 45 52 56 4F :
>18C1 52 47 41 4E 47 3A 0D 0D :
>18C9 20 20 31 2E 20 46 4F 52 :
>18D1 54 4C 41 55 46 45 4E 44 :
>18D9 0D 20 20 32 2E 20 45 49 :
>18E1 4E 5A 45 4C 4E 0D 0D 0D :
>18E9 00 0D 0D 0D 20 31 2E 20 :
>18F1 44 49 52 45 43 54 4F 52 :
>18F9 59 0D 20 32 2E 20 56 41 :
>1901 4C 49 44 49 45 52 45 4E :
>1909 0D 20 33 2E 20 46 4F 52 :
>1911 4D 41 54 49 45 52 45 4E :
>1919 0D 00 0D 0D 20 56 41 4C :
>1921 49 44 49 45 52 45 4E 20 :
>1929 2E 2E 2E 0D 00 0D 0D 20 :
>1931 12 20 51 55 45 4C 4C 2D :
>1939 44 49 53 4B 45 54 54 45 :
>1941 20 45 49 4E 4C 45 47 45 :
>1949 4E 20 92 0D 00 0D 0D 20 :
>1951 52 45 41 44 49 4E 47 20 :
>1959 00 0D 0D 20 12 20 5A 49 :
>1961 45 4C 2D 44 49 53 4B 45 :
>1969 54 54 45 20 45 49 4E 4C :
>1971 45 47 45 4E 20 92 0D 00 :
>1979 0D 0D 20 57 52 49 54 49 :
>1981 4E 47 20 00 0D 0D 20 20 :
>1989 12 20 4B 20 4F 20 50 20 :
>1991 49 20 45 20 20 20 46 20 :
>1999 45 20 52 20 54 20 49 20 :
>19A1 47 20 21 20 92 0D 00 0D :
>19A9 0D 20 20 44 49 53 4B 4E :
>19B1 41 4D 45 4E 20 55 4E 44 :
>19B9 20 49 44 20 45 49 4E 47 :
>19C1 45 42 45 4E 0D 00 0D 0D :
>19C9 0D 20 4E 4F 43 48 20 45 :
>19D1 49 4E 20 56 45 52 53 55 :
>19D9 43 48 20 3F 0D 00 0D 0D :
>19E1 20 20 31 2E 20 55 45 42 :
>19E9 45 52 53 50 52 49 4E 47 :
>19F1 45 4E 20 3F 0D 20 20 32 :
>19F9 2E 20 4E 45 55 45 52 20 :
>1A01 56 45 52 53 55 43 48 20 :
>1A09 3F 0D 00 0D 0D 20 4E 41 :
>1A11 45 43 48 53 54 45 53 20 :
>1A19 46 49 4C 45 20 3A 00 0D :
>1A21 20 20 12 20 53 43 52 41 :
>1A29 54 43 48 2D 44 49 53 4B :
>1A31 20 45 49 4E 4C 45 47 45 :
>1A39 4E 20 92 0D 00 0D 0D 20 :
>1A41 12 20 53 49 43 48 45 52 :
>1A49 20 3F 20 2D 20 53 50 41 :
>1A51 43 45 2C 20 53 4F 4E 53 :
>1A59 54 20 5C 20 92 0D 00 0D :
>1A61 0D 20 53 43 52 41 54 43 :
>1A69 48 49 4E 47 20 00 20 42 :
>1A71 4C 4F 45 43 4B 45 20 5A :
>1A79 55 20 4B 4F 50 49 45 52 :
>1A81 45 4E 20 92 0D 00 20 42 :
>1A89 4C 4F 45 43 4B 45 20 5A :
>1A91 55 20 53 43 52 41 54 43 :
>1A99 48 45 4E 20 92 0D 00 0D :
>1AA1 20 2A 2A 2A 20 53 50 41 :
>1AA9 43 45 20 2A 2A 2A 0D 20 :
>1AB1 20 46 55 45 52 20 57 45 :
>1AB9 49 54 45 52 0D 0D 20 12 :
>1AC1 20 42 49 54 54 45 20 57 :
>1AC9 41 45 48 4C 45 4E 20 53 :
>1AD1 49 45 20 92 0D 00 20 34 :
>1AD9 2E 20 4E 4F 43 48 4D 41 :
>1AE1 4C 20 4B 4F 50 49 45 52 :
>1AE9 45 4E 0D 00 24 00 49 00 :
>1AF1 2C 58 2C 52 00 2C 58 2C :
>1AF9 57 00 00 00 00 00 00 00 :
```

Listing 1. Geben Sie das Programm »Super Copy« bitte mit dem TEDMON ein

Formatieren in 30 Sekunden

Schnelles Formatieren ist auch vom VC 20 aus möglich. Die neue Routine ist fast dreimal schneller als die des Betriebssystems. Dabei berichtigt das Programm auch einen kleinen Schönheitsfehler des DOS.

Die Formatierungs-Routine der Floppy 1541 ist, vor allem bei häufiger Anwendung, nervtötend langsam. Doch mit wenig Aufwand läßt sich eine etwa dreimal schnellere Formatierungs-Routine ins Betriebssystem des VC 20 einbauen.

Der Trick an der Sache ist, daß anstelle der Berechnung des Sektoren-Abstandes Erfahrungswerte genommen werden. So spart man fast zwei Drittel der Zeit gegenüber dem normalen Formatierungs-Vorgang.

Dafür verzichtet die Routine nicht auf das notwendige Verify, da das Formatieren die einzige Möglichkeit bietet, defekte Disketten noch vor dem ersten Datenverlust zu erkennen und gegebenenfalls auszusortieren. Dazu benötigt das Verifizieren nur wenige Sekunden, die Datensicherheit geht vor.

So quasi nebenbei wird noch ein »Schönheitsfehler« des DOS korrigiert. Jeder Block wird mit 256 Nullen gefüllt, anstelle eines $4B, gefolgt von 255 $01-Byte. Geben Sie das »Disk-Format-System« (siehe Listing) bitte ein und speichern Sie es. Es wird mit RUN aktiviert und erzeugt dann auf der Diskette ein File namens »DISK-FORMAT VC 20«. Dies ist dann das lauffähige Programm, das Sie laden und ebenfalls mit RUN starten. Die nun ins Betriebssystem eingebundene Routine belegt den Bereich ab $B000. Zum Auslösen der Routine geben Sie einfach SAVE ohne Filenamen ein. Daraufhin fragt das Programm nach einem maximal 16stelligen Diskettennamen und einer zweistelligen ID. Schließlich können Sie noch den ersten und letzten zu formatierenden Track angeben. Die Eingabe muß hexadezimal erfolgen und darf von $01 bis $29 gehen.

Vorsicht! Wird eine größere Zahl als $29 eingegeben, kann der Schreib-/Lesekopf am oberen Ende anschlagen. Auch beim Formatieren von einzelnen Tracks wird das Directory gelöscht. Beachten Sie, daß eine geänderte ID einen »DISK ID MISMATCH ERROR« hervorruft. Das Programm funktioniert nur mit einer 8-KByte-Erweiterung im $A000-Bereich. Am besten, Sie verwenden eine schaltbare Erweiterung (mit DIP-Schaltern auf der Platine).

(K.-H. Templin/og)

```
0 DATA14,18,10,0,158,32,52,54,50,52,32,3
2,0,0,0,162,64,160,18,134,2,132
1 DATA3,162,0,160,176,134,4,132,5,160,0,
162,5,177,2,145,4,200,208,249,230
2 DATA3,230,5,202,208,242,120,169,242,14
1,50,3,169,179,141,51,3,88,96,234
3 DATA234,165,10,201,36,144,7,169,18,133
,67,76,19,5,32,75,242,133,67,169
4 DATA0,133,27,160,0,162,0,165,57,153,0,
3,200,200,165,27,153,0,3,200,165
5 DATA10,153,0,3,200,165,19,153,0,3,200,
165,18,153,0,3,200,169,15,153,0
6 DATA3,200,153,0,3,200,169,0,89,250,2,8
9,251,2,89,252,2,89,253,2,153,249
7 DATA2,230,27,165,27,197,67,144,190,169
,3,133,49,152,72,138,157,0,7,232
8 DATA208,250,32,48,254,104,168,136,32,2
29,253,32,245,253,169,7,133,49
9 DATA32,233,245,133,58,32,143,247,169,0
,133,50,32,14,254,169,255,141,1
10 DATA28,162,5,80,254,184,202,208,250,1
62,10,164,50,80,254,184,185,0,3
11 DATA141,1,28,200,202,208,243,162,9,80
,254,184,169,85,141,1,28,202,208
12 DATA245,169,255,162,5,80,254,184,141,
1,28,202,208,247,162,187,80,254
13 DATA184,189,0,1,141,1,28,232,208,244,
160,0,80,254,184,177,48,141,1,28
14 DATA200,208,245,169,85,162,8,80,254,1
84,141,1,28,202,208,247,165,50,24
15 DATA105,10,133,50,198,27,208,149,80,2
54,184,80,254,184,32,0,254,169,200
16 DATA133,31,169,0,133,48,169,3,133,49,
165,67,133,27,32,86,245,162,10,160
17 DATA0,80,254,184,173,1,28,209,48,208,
14,200,202,208,242,24,165,48,105
18 DATA10,133,48,76,53,6,198,31,208,209,
169,6,76,211,253,32,86,245,160,187
19 DATA80,254,184,173,1,28,217,0,1,208,2
31,200,208,242,162,252,80,254,184
20 DATA173,1,28,217,0,7,208,215,200,202,
208,241,198,27,208,176,76,158,253
21 DATA160,0,185,224,6,153,0,2,200,204,2
23,6,144,244,173,223,6,141,116,2
22 DATA173,222,6,141,123,2,169,0,133,127
,32,0,193,172,123,2,185,0,2,133
23 DATA18,185,1,2,133,19,32,7,211,169,26
,141,5,28,169,192,133,0,165,0,48
24 DATA252,174,220,6,134,10,169,224,133,
2,165,2,48,252,201,2,176,12,232
25 DATA236,221,6,144,236,32,64,238,96,23
4,234,162,2,76,10,230,0,0,0,0,0
26 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,1,41,12,14,84,82,65
27 DATA67,75,32,48,49,45,52,48,44,51,54,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
28 DATA0,0,162,0,32,135,178,160,0,32,207
,255,201,13,240,8,153,224,177,200
29 DATA192,16,144,241,169,44,153,224,177
,200,140,222,177,162,71,32,135,178
30 DATA162,0,32,207,255,201,13,240,9,153
,224,177,200,232,224,2,144,240,140
31 DATA223,177,162,83,32,135,178,32,207,
255,133,250,32,207,255,133,251,169
```

Listing. »Disk-Format-System«

```
32 DATA0,133,208,162,98,32,135,178,32,20
7,255,133,252,32,207,255,133,253
33 DATA169,0,133,208,165,250,166,251,32,
4,180,141,220,177,165,252,166,253
34 DATA32,4,180,141,221,177,238,221,177,
234,234,234,234,234,234,234,234
35 DATA234,234,234,234,234,234,76,147,17
8,189,77,179,240,6,32,210,255,232
36 DATA208,245,96,169,13,32,210,255,169,
13,32,210,255,169,0,162,176,133
37 DATA167,134,168,169,0,162,5,133,169,1
34,170,169,8,32,177,255,169,111
38 DATA32,147,255,169,77,32,168,255,169,
45,32,168,255,169,87,32,168,255
39 DATA160,0,165,169,32,168,255,165,170,
32,168,255,169,30,32,168,255,177
40 DATA167,32,168,255,200,192,30,144,246
,32,174,255,24,165,167,105,30,133
41 DATA167,144,3,230,168,24,165,169,166,
170,105,30,133,169,144,2,230,170
42 DATA224,7,144,173,201,0,144,169,169,8
,32,177,255,169,111,32,147,255,169
43 DATA77,32,168,255,169,45,32,168,255,1
69,69,32,168,255,169,96,32,168,255
44 DATA169,6,32,168,255,32,174,255,169,0
,133,144,169,8,32,180,255,169,111
45 DATA32,150,255,32,165,255,32,210,255,
36,144,80,246,32,171,255,76,220
46 DATA179,0,0,0,0,0,147,42,32,68,73,83,
75,45,70,79,82,77,65,84,45,83,89
47 DATA83,84,69,77,32,42,13,32,32,32,40,
67,41,32,49,57,56,53,32,66,89,32
48 DATA75,79,83,83,13,13,13,68,73,83,75,
78,65,77,69,58,32,0,0,0,0,0,0
49 DATA0,0,0,0,0,0,0,0,13,13,68,73,83,75
,45,73,68,58,32,0,13,13,70,82,79
50 DATA77,32,84,82,65,67,75,58,36,0,13,1
3,84,79,32,84,82,65,67,75,58,36
51 DATA0,13,13,65,78,79,84,72,69,82,32,7
0,79,82,77,65,84,32,40,89,47,78
52 DATA41,32,63,32,13,13,0,0,0,0,0,32,41
,180,162,111,32,135,178,32,228,255
53 DATA240,251,201,89,208,3,76,0,178,96,
0,165,183,240,3,76,133,246,32,0
54 DATA178,169,1,162,0,160,0,24,96,133,2
,134,3,165,2,201,65,144,3,24,105
55 DATA9,41,15,10,10,10,10,133,2,165,3,2
01,65,144,3,24,105,9,41,15,5,2,133
56 DATA2,96,169,242,141,50,3,169,179,141
,51,3,96
100 OPEN8,8,8,"DISK-FORMAT VC20,P,W"
110 PRINT#8,CHR$(1)CHR$(18);
120 FORI=0TO1138:READA:P=P+A:PRINT#8,CHR
$(A);:NEXT
130 CLOSE8
140 IFP<>124284THENPRINT"DATAS NICHT OK"
:STOP
150 PRINT"DATAS OK"
```

Listing. »Disk-Format-System«. Bitte beachten Sie die Eingabehinweise auf Seite 130.

Schnell kopiert mit Hypra-Copy VC 20

Hypra-Copy VC 20 ist ein schnelles und komfortables Filecopy-Programm für den VC 20. Das Kopieren wird um das Vier- bis Fünffache beschleunigt.

Trotz der aufwendigen Lade- und Speicherroutinen ist es gelungen, das Programm insgesamt recht kurz zu halten; Hypra-Copy belegt, gespeichert auf der Diskette, nur ganze 15 Blöcke. Somit ist der verfügbare Arbeitsspeicher sehr groß. Hat man das Programm (siehe Listing) abgetippt und gespeichert, wird nach dem Start das eigentliche Programm »Hypra-Copy VC 20« als File auf Diskette erzeugt. So kann es später ganz normal mit »LOAD "HYPRA-COPY VC 20",8« geladen und mit »RUN« gestartet werden. Es erscheint dann das Hauptmenü:

HYPRA-COPY
:C: Copy Files
:S: Scratch Files
:D: Directory
:O: Order Disk

Die Bedienung ergibt sich damit eigentlich schon von selbst. Drückt man die Taste <O>, erscheint auf dem Bildschirm eine eckige Klammer mit einem blinkenden Cursor dahinter.

Nun kann man den Befehl eingeben, der zur Floppy geschickt werden soll. Reagiert die Floppy mit einer Fehlermeldung, so wird diese auf dem Bildschirm ausgegeben.

Bei Betätigung der Taste <D> erscheint das Directory. Der Ausdruck kann durch die <CTRL>-Taste angehalten werden. Durch Drücken der <S>-Taste gelangt man in den Scratch-Modus (löschen von Files). Um Verwechslungen mit dem Copy-Modus auszuschließen, wird erst einmal mit einer dicken, reversen Balkenüberschrift darauf aufmerksam gemacht, daß man sich tatsächlich im Scratch-Modus befindet. Danach erscheint das Directory, jedoch erscheint hinter jedem Filenamen ein »(Y/N)«; relative Files sowie Files, deren Länge gleich 0 ist, werden bei dieser Befragung übergangen, denn diese Routine wird auch beim Kopieren gebraucht, und Files mit diesen Eigenschaften kann Hypra-Copy nicht kopieren.

Files, die sich Hypra-Copy merken soll, müssen mit <Y> markiert werden. Ist man versehentlich im Scratch-Modus gelandet, so kann man diesen Modus mit der STOP-Taste verlassen (ansonsten dient die STOP-Taste dazu, Hypra-Copy zu beenden; man kann es dann aber wieder mit »RUN« starten).

So werden nach und nach alle Files des Directories durchgegangen. Will man nur einige wenige Files am Anfang eines ellenlangen Directories löschen (oder kopieren), so braucht

man die restlichen Files nicht mehr mit <N> zu bearbeiten, es genügt ein Druck auf die <↑>-Taste, und man gelangt zum Ende des Directories. Hier wird dann gefragt, ob man nun auch sicher ist, daß diese Files gelöscht werden sollen. Beantwortet man diese Frage mit <N>, gelangt man zurück in das Hauptmenü, andernfalls werden die markierten Files gelöscht.

Damit man weiß, wie weit das Programm mit dem Löschen ist, wird immer der Name des Files ausgegeben, das gerade gelöscht wird. Am Ende des Löschvorgangs gelangt man automatisch wieder in das Hauptmenü.

Der Copy-Modus

So, nun zum Copy-Modus, in den man mit der <C>-Taste kommt. Zuerst werden wieder, genau wie beim Löschen, die Namen der zu kopierenden Files eingelesen.

Danach wird gefragt, ob man die Files einzeln oder gesammelt kopieren will und ob beim Speichern ein »VERIFY« durchgeführt werden soll. Verzichtet man auf dieses Verify, so wird etwas schneller kopiert. Um den Bedienungskomfort zu erhöhen, werden die Antworten auf diese Fragen, wie auch auf die »SURE?«-Frage (sure = Sind Sie sicher?) im Scratch-Modus und die »SAVE BUFFER AGAIN?«-Frage (Soll das gleiche noch einmal gespeichert werden?), nicht per GET ($FFE4), sondern per BASIN ($FFCF) eingelesen, wobei die erwartete Antwort schon vorgegeben ist; man braucht also nur noch RETURN drücken; natürlich kann diese vorgegebene Antwort aber auch überschrieben werden.

Nun werden die Programme geladen. Ist die Summe der Blöcke der einzelnen Files kleiner als der im VC 20 verfügbare Speicher, können sogar alle auf einmal eingelesen werden; andernfalls wird ein erneuter Ladeanlauf nötig.

Sind die Files geladen, wird man aufgefordert, die Ziel-Disk einzulegen; außerdem wird noch die Anzahl der zu speichernden Blocks angegeben, und ein Untermenü erscheint.

Es besteht jetzt noch die Möglichkeit, diverse Directories anzusehen oder Befehle zur Floppy zu schicken.

Man kann also in aller Ruhe eine passende Diskette aussuchen. Ist man gewillt fortzufahren, so drückt man einfach die Funktionstaste <F7>, dann beginnt nämlich das Speichern. Anschließend wird man gefragt, ob man eben diese Files noch einmal auf eine andere Diskette schreiben will. Beantwortet man diese Frage mit <Y>, beginnt der komplette SAVE-Vorgang von neuem. Andernfalls wird normal fortgefahren und, wie schon erwähnt, können erneute Ladeanläufe erforderlich werden.

Wurden alle Files kopiert, so wird in das Hauptmenü zurückgekehrt.

Nun zur Fehlerbehandlung: Mit »Fehlern« sind nicht Fehler gemeint, die in der Programmstruktur von Hypra-Copy liegen, sondern Fehler, die von der Floppy signalisiert werden; sei es, daß man einen unkorrekten Befehl an die Floppy schickt, man beim Kopieren die falsche Diskette einlegt oder Schreib- oder Lesefehler auftreten.

Erste Hilfe bei Fehlern

In solchen oder ähnlichen Fällen besteht immer die Möglichkeit, den Vorgang, bei dem der Fehler aufgetreten ist, zu wiederholen beziehungsweise zu übergehen. Treten Fehler beim Öffnen eines Files auf, so kann man sogar nochmals Directories ansehen und Befehle an die Floppy schicken. War es nicht möglich, ein File korrekt zu laden und wurde es übergangen, so wird dies bei der Angabe der zu speichernden Blöcke, wie auch beim Speichern selbst, berücksichtigt. Können Files aus irgendeinem Grund nicht korrekt gespeichert werden, so kann man diese auch überspringen.

(Burkhard Graves/K. H. Templin/og)

```
Ø DATA 11,18,193,7,158,52,54,5Ø,49,Ø,Ø,Ø,1
  69,8,141,15,144,12Ø,169,91,141           <238>
1 DATA 24,3,238,25,3,88,162,Ø,134,5,142,59
  ,19,169,64,133,157,169,213,16Ø           <Ø98>
2 DATA 18,32,3Ø,2Ø3,32,225,19,76,28,18,165
  ,247,24,1Ø5,19,133,247,144,2,23Ø         <129>
3 DATA 248,2Ø1,234,96,169,176,133,247,133,
  249,169,32,133,248,133,25Ø,96            <213>
4 DATA 13,18,32,33,32,83,67,82,65,84,67,72
  ,32,7Ø,73,76,69,83,32,33,32,17           <131>
5 DATA 13,Ø,13,17,67,79,8Ø,89,32,83,69,8Ø,
  65,82,65,84,69,76,89,32,63,32            <222>
6 DATA 78,157,Ø,13,17,86,69,82,73,7Ø,89,32
  ,63,32,78,157,Ø,13,17,83,85,82           <135>
7 DATA 69,32,63,32,89,157,Ø,13,17,58,84,58
  ,32,84,82,89,32,65,71,65,73,78           <Ø1Ø>
8 DATA 13,17,58,7Ø,55,58,32,84,79,32,67,79
  ,78,84,73,78,85,69,17,13,Ø,13            <175>
9 DATA 17,83,65,86,69,32,66,85,7Ø,7Ø,69,82
  ,32,65,71,65,73,78,32,63,32,78           <1ØØ>
1Ø DATA 157,Ø,147,5,17,29,62,32,72,89,8Ø,8
   2,65,45,67,79,8Ø,89,32,6Ø,13,29         <ØØ7>
11 DATA 29,196,196,196,196,196,196,196,196
   ,196,196,196,196,17,17,13,58,67         <Ø53>
12 DATA 58,32,67,79,8Ø,89,32,7Ø,73,76,69,8
   3,17,13,58,83,58,32,83,67,82,65         <234>
13 DATA 84,67,72,32,7Ø,73,76,69,83,13,17,5
   8,68,58,32,68,73,82,69,67,84,79         <Ø82>
14 DATA 82,89,17,13,58,79,58,32,79,82,68,6
   9,82,32,68,73,83,75,17,13,Ø,145         <2Ø8>
15 DATA 58,7Ø,55,58,32,84,79,32,67,79,78,8
   4,73,78,85,69,17,13,Ø,18,4Ø,89          <25Ø>
16 DATA 47,78,41,146,Ø,2Ø,2Ø,2Ø,2Ø,2Ø,157,
   6Ø,45,Ø,13,17,76,73,83,84,32,7Ø         <16Ø>
17 DATA 85,76,76,32,33,17,13,Ø,13,73,78,83
   ,69,82,84,32,83,79,85,82,67,69          <17Ø>
18 DATA 32,68,73,83,75,13,Ø,13,73,78,83,69
   ,82,84,32,84,65,82,71,69,84,32          <Ø56>
19 DATA 68,73,83,75,32,195,13,Ø,32,66,76,7
   9,67,75,83,32,84,79,32,83,65,86         <Ø5Ø>
2Ø DATA 69,13,Ø,13,17,17,68,73,83,75,32,69
   ,82,82,79,82,13,Ø,13,17,17,76,79        <159>
21 DATA 65,68,32,69,82,82,79,82,32,33,Ø,32
   ,66,18,16Ø,Ø,152,145,247,2ØØ,2Ø8        <146>
22 DATA 251,23Ø,248,166,248,224,42,2Ø8,243
   ,96,169,Ø,133,198,32,228,255,24Ø        <1Ø5>
23 DATA 251,166,5,24Ø,3,76,2Ø1,2Ø,2Ø1,67,2
   Ø8,74,32,2Ø5,19,162,Ø,134,3,32          <121>
24 DATA 66,18,32,52,21,162,Ø,189,176,32,2Ø
   8,1,96,169,1Ø3,16Ø,18,32,3Ø,2Ø3         <154>
25 DATA 169,Ø,133,198,133,65,32,2Ø7,255,2Ø
   1,89,2Ø8,2,133,65,169,76,141,169        <121>
26 DATA 31,169,126,16Ø,18,32,3Ø,2Ø3,169,Ø,
   133,198,32,2Ø7,255,2Ø1,89,2Ø8,5         <Ø72>
27 DATA 169,44,141,169,31,76,2,23,2Ø1,83,2
   4Ø,3,76,2Ø1,2Ø,169,164,16Ø,19,32        <Ø55>
28 DATA 45,21,169,79,16Ø,18,32,3Ø,2Ø3,169,
   Ø,133,3,32,2Ø5,19,32,66,18,32,52        <Ø31>
29 DATA 21,32,66,18,169,Ø,133,198,169,14Ø,
   16Ø,18,32,3Ø,2Ø3,32,2Ø7,255,2Ø1         <2ØØ>
3Ø DATA 89,24Ø,1,96,169,13,32,21Ø,255,32,2
   1Ø,255,169,8,32,177,255,169,111         <ØØØ>
31 DATA 32,147,255,16Ø,Ø,177,247,24Ø,53,16
   9,83,32,168,255,32,21Ø,255,169          <113>
32 DATA 58,32,168,255,32,21Ø,255,2ØØ,177,2
   47,2Ø1,44,24Ø,9,32,168,255,32,21Ø       <224>
33 DATA 255,76,159,2Ø,169,13,32,168,255,32
   ,21Ø,255,169,8,32,174,255,32,2Ø1        <17Ø>
```

```
34 DATA 22,32,52,18,2Ø8,187,169,8,76,174,2
   55,2Ø1,136,2Ø8,1,96,2Ø1,84,2Ø8           <11Ø>
35 DATA 1,96,2Ø1,68,2Ø8,14,162,25,134,3,32
   ,52,21,166,2Ø3,224,64,24Ø,25Ø,96         <126>
36 DATA 2Ø1,79,2Ø8,49,169,93,32,21Ø,255,16
   9,8,32,177,255,169,111,32,147,255        <Ø1Ø>
37 DATA 169,Ø,133,198,32,2Ø7,255,32,168,25
   5,2Ø1,13,2Ø8,246,32,21Ø,255,169          <Ø7Ø>
38 DATA 8,32,174,255,32,2Ø1,22,176,1,96,32
   ,1Ø5,22,144,2Ø8,96,2Ø1,3,2Ø8,3           <Ø44>
39 DATA 76,187,22,76,229,19,169,13,32,21Ø,
   255,16Ø,22,169,195,32,21Ø,255,136        <Ø97>
4Ø DATA 2Ø8,25Ø,96,32,71,22,169,96,133,185
   ,169,1,162,2Ø8,16Ø,255,32,141,22         <19Ø>
41 DATA 144,3,76,96,22,165,186,32,18Ø,255,
   165,185,32,15Ø,255,169,Ø,133,144         <Ø28>
42 DATA 16Ø,3,32,79,22,133,2,32,79,22,136,
   2Ø8,245,132,5,166,2,32,2Ø5,221           <Ø19>
43 DATA 169,32,32,21Ø,255,32,79,22,164,3,2
   Ø8,66,2Ø1,34,2Ø8,62,133,5,32,21Ø         <Ø41>
44 DATA 255,16Ø,19,169,Ø,133,6,145,247,136
   ,16,251,2ØØ,32,79,22,32,21Ø,255          <Ø96>
45 DATA 2Ø1,34,24Ø,5,2ØØ,145,247,2Ø8,241,1
   69,44,2ØØ,145,247,32,79,22,2Ø1           <172>
46 DATA 8Ø,24Ø,8,2Ø1,83,24Ø,4,2Ø1,85,2Ø8,5
   ,2ØØ,145,247,2Ø8,5,32,21Ø,255,2Ø8        <2Ø5>
47 DATA 231,17Ø,24Ø,5,32,21Ø,255,2Ø8,175,1
   65,2,24Ø,115,2Ø1,92,176,111,165          <Ø14>
48 DATA 5,24Ø,1Ø7,169,29,133,211,169,8Ø,16
   Ø,19,32,3Ø,2Ø3,169,Ø,133,198,133         <ØØ8>
49 DATA 251,32,228,255,24Ø,251,2Ø1,3,2Ø8,8
   ,32,71,22,1Ø4,1Ø4,76,28,18,2Ø1           <242>
5Ø DATA 94,2Ø8,5,133,6,76,36,22,2Ø1,89,2Ø8
   ,36,169,157,141,93,19,32,25,22           <Ø46>
51 DATA 16Ø,Ø,165,2,145,247,32,52,18,2Ø8,4
   1,169,97,16Ø,19,32,3Ø,2Ø3,76,71          <2Ø3>
52 DATA 22,169,88,16Ø,19,76,3Ø,2Ø3,2Ø1,78,
   2Ø8,186,169,Ø,141,93,19,32,25,22         <187>
53 DATA 16Ø,Ø,152,145,247,165,6,24Ø,3,76,7
   1,22,169,13,32,21Ø,255,173,141           <131>
54 DATA 2,2Ø8,251,16Ø,2,76,89,21,169,8,32,
   195,255,76,231,255,32,165,255,166        <248>
55 DATA 144,24Ø,18,1Ø4,1Ø4,32,71,22,32,2Ø1
   ,22,144,8,32,1Ø5,22,176,3,76,52          <226>
56 DATA 21,96,169,152,16Ø,18,32,3Ø,2Ø3,16Ø
   ,Ø,132,198,32,228,255,17Ø,169,Ø          <Ø15>
57 DATA 224,84,2Ø8,2,24,96,224,136,2Ø8,2,5
   6,96,224,3,2Ø8,234,76,187,22,32          <Ø21>
58 DATA 189,255,32,89,246,169,8,133,184,13
   3,186,169,8,32,177,255,169,111           <171>
59 DATA 32,147,255,169,85,32,168,255,169,7
   3,32,168,255,169,8,32,174,255,32         <223>
6Ø DATA 192,255,176,3,76,2Ø1,22,96,32,71,2
   2,32,82,253,169,13,32,21Ø,255,76         <237>
61 DATA 129,227,169,8,32,18Ø,255,169,111,3
   2,15Ø,255,32,165,255,17Ø,56,233          <ØØ4>
62 DATA 48,133,14Ø,24Ø,1Ø,138,72,169,174,1
   6Ø,19,32,3Ø,2Ø3,1Ø4,166,14Ø,24Ø          <212>
63 DATA 3,32,21Ø,255,32,165,255,2Ø1,13,2Ø8
   ,242,169,8,32,171,255,24,165,14Ø         <136>
64 DATA 24Ø,1,56,96,32,66,18,169,Ø,133,139
   ,169,Ø,133,174,133,6,168,153,83          <153>
65 DATA 32,2ØØ,192,9Ø,2Ø8,248,169,36,133,1
   75,16Ø,Ø,177,247,2Ø8,1,96,32,36          <Ø45>
66 DATA 21,165,139,24Ø,16,169,114,16Ø,19,3
   2,3Ø,2Ø3,169,Ø,133,198,32,228,255        <Ø96>
67 DATA 24Ø,251,169,93,133,64,133,253,23Ø,
   139,16Ø,Ø,177,247,133,254,24Ø,114        <23Ø>
68 DATA 197,64,144,6,32,188,23,76,9,23,162
   ,Ø,134,185,165,64,56,229,254,133         <Ø41>
69 DATA 64,165,253,56,229,254,133,253,165,
   174,133,1Ø5,165,175,133,1Ø6,165          <144>
7Ø DATA 1Ø5,133,174,165,1Ø6,133,175,32,92,
   24,176,17,32,2Ø4,24,144,23,169           <181>
71 DATA 189,16Ø,19,32,3Ø,2Ø3,32,1Ø5,22,144
   ,226,165,253,24,1Ø1,254,133,253          <126>
72 DATA 169,Ø,24Ø,2,169,234,133,255,164,6,
   165,175,153,83,32,2ØØ,165,174,153        <Ø46>
73 DATA 83,32,2ØØ,165,255,153,83,32,2ØØ,13
   2,6,32,52,18,24Ø,6,165,65,2Ø8,148        <123>
74 DATA 24Ø,134,169,Ø,133,193,133,6,169,36
   ,133,194,165,249,133,247,165,25Ø         <13Ø>
75 DATA 133,248,32,36,21,169,135,16Ø,19,32
   ,3Ø,2Ø3,169,93,56,229,253,17Ø,169        <233>
76 DATA Ø,32,2Ø5,221,169,157,16Ø,19,32,3Ø,
   2Ø3,169,13,141,59,19,133,5,169           <Ø51>
77 DATA 27,16Ø,19,32,3Ø,2Ø3,32,225,19,2Ø1,
   136,2Ø8,235,164,6,185,83,32,24Ø          <152>
78 DATA 56,133,175,2ØØ,185,83,32,133,174,2
   ØØ,185,83,32,2ØØ,132,6,17Ø,24Ø           <238>
79 DATA 25,169,1,133,185,32,92,24,176,16,3
   2,131,28,32,71,22,32,2Ø1,22,144          <12Ø>
8Ø DATA 5,32,1Ø5,22,144,235,165,174,133,19
   3,165,175,133,194,32,52,18,2Ø8           <112>
81 DATA 193,169,188,16Ø,18,32,3Ø,2Ø3,169,Ø
   ,133,198,32,2Ø7,255,2Ø1,89,2Ø8           <2Ø8>
82 DATA 3,76,188,23,165,247,133,249,165,24
   8,133,25Ø,96,162,5,165,185,41,1          <186>
83 DATA 2Ø8,7,162,32,16Ø,73,76,1Ø9,24,16Ø,
   81,32,23Ø,241,138,32,21Ø,255,166         <Ø19>
84 DATA 247,164,248,232,2Ø8,1,2ØØ,169,18,3
   2,141,22,176,12,234,234,234,234          <125>
85 DATA 234,234,234,234,234,234,24,96,32,7
   1,22,165,185,133,254,169,Ø,141           <1Ø3>
86 DATA 59,19,169,13,32,21Ø,255,133,5,169,
   27,16Ø,19,32,3Ø,2Ø3,169,145,32           <131>
87 DATA 21Ø,255,32,21Ø,255,169,152,16Ø,18,
   32,3Ø,2Ø3,32,225,19,2Ø1,84,2Ø8           <19Ø>
88 DATA 7,165,254,133,185,76,92,24,2Ø1,136
   ,2Ø8,2Ø9,56,96,165,186,32,18Ø,255        <21Ø>
89 DATA 165,185,32,15Ø,255,16Ø,Ø,32,165,25
   5,234,234,234,234,145,174,234,234        <138>
9Ø DATA 2ØØ,192,2,2Ø8,24Ø,32,71,22,165,174
   ,24,1Ø5,2,133,174,144,11,23Ø,175         <185>
91 DATA 2Ø8,7,165,167,2Ø1,8,24Ø,23,Ø,169,1
   ,133,167,169,Ø,133,144,165,167           <136>
92 DATA 32,23,238,169,111,32,192,238,165,1
   44,16,227,23Ø,167,165,167,2Ø1,16         <225>
93 DATA 2Ø8,23Ø,169,141,162,26,133,167,134
   ,168,169,Ø,162,3,133,169,134,17Ø         <143>
94 DATA 169,8,32,23,238,169,111,32,192,238
   ,169,77,32,228,238,169,45,32,228         <Ø38>
95 DATA 238,169,87,32,228,238,16Ø,Ø,165,16
   9,32,228,238,165,17Ø,32,228,238          <185>
96 DATA 169,3Ø,32,228,238,177,167,32,228,2
   38,2ØØ,192,3Ø,144,246,32,4,239           <Ø12>
97 DATA 24,165,167,1Ø5,3Ø,133,167,144,3,23
   Ø,168,24,165,169,166,17Ø,1Ø5,3Ø          <249>
98 DATA 133,169,144,2,23Ø,17Ø,224,5,144,17
   3,2Ø1,Ø,144,169,169,8,32,23,238          <1ØØ>
99 DATA 169,111,32,192,238,169,77,32,228,2
   38,169,45,32,228,238,169,69,32           <128>
1ØØ DATA 228,238,169,139,32,228,238,169,4,
    32,228,238,32,1,239,234,234,234         <142>
1Ø1 DATA 234,234,234,234,12Ø,169,1,133,167
    ,16Ø,255,32,48,26,192,255,24Ø,68        <Ø58>
1Ø2 DATA 234,234,234,234,162,2,165,167,24Ø
    ,2,162,4,173,Ø,35,2Ø8,7,238,1,35        <233>
1Ø3 DATA 173,1,35,44,169,Ø,133,168,189,Ø,3
    5,145,174,23Ø,174,2Ø8,2,23Ø,175         <222>
1Ø4 DATA 232,228,168,2Ø8,24Ø,162,Ø,134,167
    ,234,234,234,234,173,Ø,35,2Ø8,19Ø       <Ø39>
1Ø5 DATA 234,234,234,234,234,169,64,133,14
    4,24,96,56,96,169,128,141,31,145        <249>
1Ø6 DATA 173,31,145,41,2,24Ø,249,169,Ø,141
    ,31,145,162,6,2Ø2,2Ø8,253,162,4         <Ø1Ø>
1Ø7 DATA 173,31,145,1Ø6,1Ø2,176,1Ø6,1Ø2,17
    6,234,234,2Ø2,2Ø8,242,165,176,73        <217>
1Ø8 DATA 255,96,32,5,26,2Ø1,255,24Ø,248,16
    Ø,Ø,169,128,141,31,145,173,31,145       <Ø56>
1Ø9 DATA 41,2,24Ø,249,169,Ø,141,31,145,162
    ,7,2Ø2,2Ø8,253,173,31,145,1Ø6,1Ø2       <146>
11Ø DATA 176,1Ø6,1Ø2,176,234,234,234,234,2
    34,173,31,145,1Ø6,1Ø2,176,1Ø6,1Ø2       <159>
111 DATA 176,234,234,234,234,234,173,31,14
    5,1Ø6,1Ø2,176,1Ø6,1Ø2,176,234,234       <Ø21>
112 DATA 234,234,234,173,31,145,1Ø6,1Ø2,17
    6,1Ø6,1Ø2,176,165,176,73,255,153        <138>
113 DATA Ø,35,2ØØ,2Ø8,173,96,165,Ø,41,6,2Ø
    1,2,24Ø,3,76,158,253,234,169,5,133      <193>
114 DATA 9,162,9Ø,134,75,162,Ø,169,82,133,
    36,32,86,245,8Ø,254,184,173,1,28        <ØØ5>
115 DATA 197,36,24Ø,9,198,75,2Ø8,239,169,1
    Ø,76,1Ø5,249,8Ø,254,184,173,1,28        <1Ø6>
```

Listing. »Hypra-Copy«. Bitte mit dem Checksummer (Seite 129) eingeben.

```
116 DATA 149,37,232,224,7,2Ø8,243,32,151,2
    44,165,22,69,23,69,24,69,25,69,26      <148>
117 DATA 24Ø,7,198,9,2Ø8,192,76,3Ø,244,165
    ,24,197,6,24Ø,3,76,11,244,133,34       <117>
118 DATA 169,6,133,49,76,6Ø,4,165,18,166,1
    9,133,22,134,23,165,6,133,24,165       <117>
119 DATA 7,133,25,169,Ø,69,22,69,23,69,24,
    69,25,133,26,32,52,249,162,9Ø,32       <11Ø>
12Ø DATA 86,245,16Ø,Ø,8Ø,254,184,173,1,28,
    217,36,Ø,24Ø,6,2Ø2,2Ø8,237,76,81       <ØØ2>
121 DATA 245,2ØØ,192,8,2Ø8,234,32,86,245,8
    Ø,254,184,173,1,28,145,48,2ØØ,2Ø8      <254>
122 DATA 245,16Ø,186,8Ø,254,184,173,1,28,1
    53,Ø,1,2ØØ,2Ø8,244,32,224,248,165      <218>
123 DATA 56,197,71,24Ø,3,76,246,244,32,233
    ,245,197,58,24Ø,3,76,2,245,16Ø,Ø       <Ø42>
124 DATA 169,85,32,82,4,185,Ø,6,133,119,44
    ,Ø,24,16,251,169,16,141,Ø,24,44        <145>
125 DATA Ø,24,48,251,162,Ø,138,1Ø2,119,42,
    42,1Ø2,119,42,42,141,Ø,24,138,1Ø2      <129>
126 DATA 119,42,42,1Ø2,119,42,42,141,Ø,24,
    138,1Ø2,119,42,42,1Ø2,119,42,42        <233>
127 DATA 141,Ø,24,138,1Ø2,119,42,42,1Ø2,11
    9,42,42,141,Ø,24,162,2,2Ø2,2Ø8,253     <112>
128 DATA 169,15,141,Ø,24,2ØØ,2Ø8,173,234,2
    34,234,234,234,234,234,173,Ø,28        <Ø57>
129 DATA 9,8,141,Ø,28,173,Ø,6,2Ø8,3,76,158
    ,253,197,24,2Ø8,249,133,6,173,1        <Ø74>
13Ø DATA 6,133,7,76,1Ø1,3,133,119,44,Ø,24,
    16,251,169,16,141,Ø,24,44,Ø,24,48      <Ø93>
131 DATA 251,162,4,169,Ø,1Ø2,119,42,42,1Ø2
    ,119,42,42,141,Ø,24,2Ø2,2Ø8,24Ø        <153>
132 DATA 234,234,234,234,234,234,169,15,14
    1,Ø,24,96,96,133,Ø,88,165,Ø,48,252     <226>
133 DATA 12Ø,96,12Ø,234,234,234,234,234,23
    4,165,24,141,Ø,6,133,6,165,25,141      <Ø12>
134 DATA 1,6,133,7,169,4,133,12Ø,169,226,3
    2,13Ø,4,2Ø1,2,144,51,16Ø,Ø,132,12Ø     <124>
135 DATA 164,12Ø,185,219,254,24Ø,18,88,32,
    118,214,12Ø,169,226,32,13Ø,4,2Ø1       <239>
136 DATA 2,144,26,23Ø,12Ø,2Ø8,231,169,192,
    32,13Ø,4,169,226,32,13Ø,4,2Ø1,2        <2Ø4>
137 DATA 144,8,169,255,32,82,4,76,34,235,1
    73,Ø,6,24Ø,248,197,24,24Ø,196,173      <188>
138 DATA Ø,6,133,6,173,1,6,133,7,76,16Ø,4,
    173,44,145,141,31,1,165,186,32,23      <13Ø>
139 DATA 238,165,185,32,192,238,169,Ø,133,
    144,32,228,238,32,228,238,32,4,239     <188>
14Ø DATA 165,144,24Ø,2,24,96,169,23,162,3Ø
    ,133,172,134,173,169,7Ø,162,1,133      <Ø46>
141 DATA 2Ø,134,21,162,5,165,186,32,23,238
    ,169,111,32,192,238,16Ø,Ø,185,15       <Ø45>
142 DATA 3Ø,32,228,238,2ØØ,192,3,2Ø8,245,1
    65,2Ø,32,228,238,165,21,32,228,238     <ØØØ>
143 DATA 169,31,32,228,238,16Ø,Ø,177,172,3
    2,228,238,2ØØ,192,31,144,246,32        <2Ø5>
144 DATA 4,239,165,172,24,1Ø5,31,133,172,1
    44,2,23Ø,173,165,2Ø,24,1Ø5,31,133      <Ø49>
145 DATA 2Ø,144,2,23Ø,21,2Ø2,2Ø8,178,165,1
    86,32,23,238,169,111,32,192,238        <Ø97>
146 DATA 16Ø,Ø,185,18,3Ø,32,228,238,2ØØ,19
    2,5,2Ø8,245,16Ø,Ø,32,4,239,12Ø,162     <Ø14>
147 DATA Ø,142,31,145,169,22Ø,141,44,145,2
    34,189,178,3Ø,157,Ø,35,232,2Ø8,247     <Ø74>
148 DATA 173,31,145,41,2,2Ø8,249,32,251,29
    ,162,Ø,189,178,31,157,Ø,35,232,2Ø8     <134>
149 DATA 247,32,251,29,32,21Ø,251,162,1,32
    ,17,253,176,23,16Ø,Ø,177,172,157       <157>
15Ø DATA 1,35,232,32,27,253,224,255,144,23
    6,32,17,253,176,3,169,255,44,169       <ØØ5>
151 DATA Ø,72,141,Ø,35,142,1,35,32,251,29,
    162,1,1Ø4,2Ø8,211,24,169,Ø,72,173      <238>
152 DATA 31,1,141,44,145,12Ø,1Ø4,96,185,Ø,
    35,133,149,162,Ø,173,31,145,41,1       <Ø11>
153 DATA 24Ø,249,173,31,145,41,2,24Ø,5,232
    ,2Ø8,246,24Ø,217,169,222,141,44        <126>
154 DATA 145,42,42,42,6,149,42,42,42,42,6,
    149,42,42,141,44,145,42,42,42,6        <189>
155 DATA 149,42,42,42,42,6,149,42,42,141,4
    4,145,42,42,42,6,149,42,42,42,42       <214>
156 DATA 6,149,42,42,141,44,145,42,42,42,6
    ,149,42,42,42,42,6,149,42,42,141       <121>
157 DATA 44,145,234,234,234,234,234,169,22
    Ø,141,44,145,2ØØ,2Ø8,148,96,16Ø        <181>
158 DATA Ø,152,89,Ø,35,2ØØ,2Ø8,25Ø,133,2Ø,
    32,142,29,136,165,2Ø,76,145,29,77      <238>
159 DATA 45,87,77,45,69,187,1,16Ø,Ø,132,17
    ,169,2,141,Ø,24,169,4,44,Ø,24,24Ø      <156>
16Ø DATA 251,169,Ø,141,Ø,24,162,4,2Ø2,2Ø8,
    253,162,1Ø,173,Ø,24,74,38,133,74       <Ø25>
161 DATA 74,38,133,97,Ø,1Ø1,Ø,173,Ø,24,74,
    38,133,74,74,38,133,97,Ø,1Ø1,Ø,173     <225>
162 DATA Ø,24,74,38,133,74,74,38,133,97,Ø,
    1Ø1,Ø,173,Ø,24,142,Ø,24,74,38,133      <Ø39>
163 DATA 74,74,38,133,165,133,145,48,69,17
    ,133,17,2ØØ,2Ø8,166,136,96,133,49      <187>
164 DATA 32,7Ø,1,177,48,72,32,74,1,1Ø4,145
    ,48,165,17,24Ø,237,76,67,232,12Ø       <163>
165 DATA 169,1Ø,141,Ø,24,162,Ø,136,2Ø8,253
    ,2Ø2,2Ø8,25Ø,169,1Ø,133,1Ø5,169        <14Ø>
166 DATA Ø,133,48,169,3,32,166,1,169,4,32,
    166,1,76,61,4,234,234,234,165,Ø        <252>
167 DATA 162,1,134,Ø,41,2,24Ø,16,166,152,1
    34,5Ø,32,46,4,32,175,3,32,3,4,76       <Ø71>
168 DATA 1Ø5,249,162,8,32,53,3,162,1Ø,32,5
    3,3,162,8,32,245,3,162,1Ø,32,245       <Ø46>
169 DATA 3,165,14Ø,48,232,76,1Ø5,249,134,5
    Ø,134,152,165,14Ø,16,97,181,131        <Ø43>
17Ø DATA 2Ø8,93,189,49,4,32,166,1,2ØØ,166,
    5Ø,169,128,149,131,165,128,149,Ø       <141>
171 DATA 165,129,149,1,177,48,24Ø,28,32,33
    ,241,166,13Ø,246,181,2Ø8,2,246,187     <141>
172 DATA 16Ø,Ø,165,128,145,48,2ØØ,165,129,
    145,48,165,128,197,34,24Ø,2,132        <165>
173 DATA 14Ø,166,5Ø,169,Ø,133,48,133,51,13
    3,46,133,54,133,12,133,8Ø,169,187      <2ØØ>
174 DATA 133,52,189,49,4,133,47,32,233,245
    ,133,58,189,5Ø,4,32,163,247,166        <Ø93>
175 DATA 5Ø,181,131,24Ø,82,32,46,4,173,Ø,2
    8,41,16,2Ø8,3,76,129,245,32,16,245     <2Ø4>
176 DATA 162,9,8Ø,254,184,2Ø2,2Ø8,25Ø,169,
    255,141,3,28,173,12,28,41,31,9,192     <243>
177 DATA 141,12,28,169,255,162,5,141,1,28,
    184,8Ø,254,184,2Ø2,2Ø8,25Ø,16Ø,187     <18Ø>
178 DATA 177,12,8Ø,254,184,141,1,28,2ØØ,2Ø
    8,245,177,48,8Ø,254,184,141,1,28       <232>
179 DATA 2ØØ,2Ø8,245,8Ø,254,76,Ø,254,96,13
    4,5Ø,44,36,4,134,152,181,131,24Ø       <2Ø6>
18Ø DATA 244,32,46,4,32,1Ø,245,16Ø,187,177
    ,12,8Ø,254,184,77,1,28,2Ø8,25,2ØØ      <252>
181 DATA 2Ø8,243,177,48,8Ø,254,184,77,1,28
    ,2Ø8,12,2ØØ,192,253,2Ø8,241,166        <219>
182 DATA 5Ø,169,Ø,149,131,96,76,197,246,18
    9,49,4,133,49,189,5Ø,4,133,13,96       <154>
183 DATA 5,1,6,4,88,32,25,241,169,132,213,
    167,24Ø,5,149,167,32,66,2Ø8,169        <Ø51>
184 DATA 64,141,249,2,169,1,133,131,32,7,2
    Ø9,144,3,76,248,2Ø7,32,62,222,246      <111>
185 DATA 181,169,Ø,133,139,133,141,169,128
    ,133,14Ø,165,128,133,6,169,224,133     <2Ø3>
186 DATA Ø,165,Ø,48,252,24Ø,36,2Ø1,1,24Ø,2
    34,165,24,133,6,165,25,133,7,162       <116>
187 DATA Ø,169,176,32,125,213,32,153,213,1
    69,226,32,125,213,32,153,213,165       <198>
188 DATA 14Ø,2Ø8,2Ø4,24Ø,2Ø6,76,35,219,42,
    48,49,42,48,5Ø,42,48,51,42,48,52       <Ø36>
189 DATA 42,48,53,42,48,54,42,48,55,42,48,
    56,42,48,57,42,49,48,42,49,49,42       <121>
19Ø DATA 49,5Ø,42,49,51,42,49,52,42,49,53,
    42,49,54,42,49,55,42,49,56,42,49       <Ø31>
191 DATA 57,42,5Ø,48,42,5Ø,49,42,5Ø,5Ø,42,
    5Ø,51,42,5Ø,52,42,5Ø,53,42,5Ø,54       <Ø55>
192 DATA 42,5Ø,55,42,5Ø,56,42,5Ø,57,42,51,
    48,Ø,Ø,Ø                               <Ø93>
2ØØ OPEN 1,8,15:CLOSE 1:IF ST=Ø THEN 24Ø   <139>
21Ø PRINT"BITTE FLOPPY ANSCHLIESSEN, DANN<
    SPACE>"                                <198>
22Ø GET T$:IF T$<>" "THEN 22Ø              <23Ø>
23Ø GOTO 2ØØ                               <166>
24Ø OPEN 8,8,8,"HYPRA-COPY VC2Ø,P,W"       <1ØØ>
25Ø PRINT#8,CHR$(1)CHR$(18);               <2Ø8>
26Ø FOR I=Ø TO 3758:READ A:P=P+A           <247>
27Ø PRINT#8,CHR$(A);                       <Ø54>
28Ø NEXT:CLOSE 8                           <ØØ6>
29Ø IF P<>4213Ø4 THEN PRINT"DATAS NICHT OK
    ":STOP                                 <Ø4Ø>
3ØØ PRINT"DATAS OK"                        <245>
```

Listing. »Hypra-Copy« (Schluß)

Schnell wie der Wind

»Hypra-Load« und »Hypra-Save« für den VC 20 mit der Floppy 1541. Laden Sie Ihre Programme mit sechsfacher Geschwindigkeit und speichern Sie sie dreimal schneller. Beide Programme in einem, das ist für Sie eine wertvolle Arbeitserleichterung.

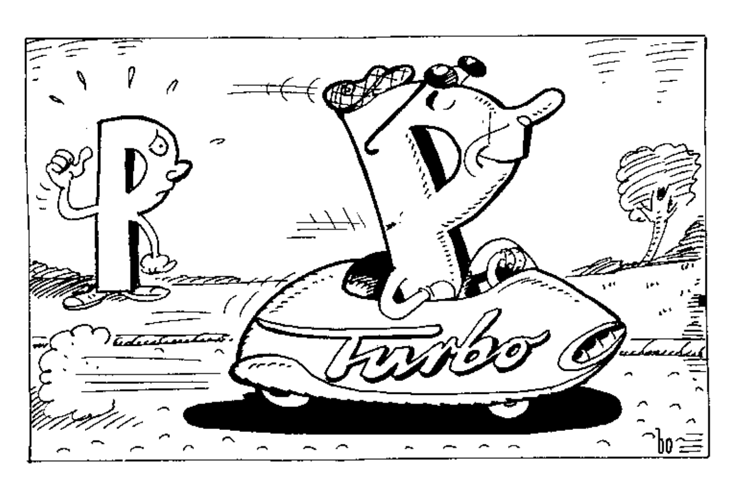
Schnelles Laden und Speichern ist nun nicht mehr dem C64 vorbehalten. Der VC 20 verfällt nunmehr in einen wahren Geschwindigkeitsrausch. Jeder, der sich intensiv mit dem VC 20 beschäftigt, wird diesen Floppy-Speeder zu schätzen wissen.

Das Prinzip der Übertragung ist dasselbe wie beim C64. Der serielle Bus wird zur Zwei-Bit-Parallelübertragung »mißbraucht«. Synchronisiert wird lediglich über die Taktfrequenz. Das Programm selbst befindet sich im Bereich ab $A000. Vor jedem Lade- oder Speichervorgang wird ein Programm in die Floppy geschrieben, das für die schnellere Übertragung verantwortlich ist.

Geben Sie das Programm »Hypra-System« (siehe Listing) am besten mit dem Checksummer VC 20 ein und speichern Sie es. Beim Starten wird auf der Diskette das Programmfile »Hypra-System VC 20« generiert.

Sind alle DATAs korrekt eingegeben, können Sie »Hypra-System VC 20« laden und durch einen Reset (oder SYS 64802) aktivieren. Einmal gestartet, läßt es sich auch durch RUN/STOP-RESTORE nicht mehr deaktivieren.

Programme, die den Bereich ab $A000 benutzen, überschreiben das »Hypra-System« und verursachen dadurch meistens einen Absturz. Der Schnelllader ist natürlich mit den meisten Programmen lauffähig. (K.-H. Templin/og)




```
Ø DATA 9,16Ø,116,16Ø,65,48,195,194,2Ø5,32,
  141,253,32,82,253,32,249,253,32             <Ø44>
1 DATA 24,229,88,32,91,228,32,164,227,32,9
  5,16Ø,32,4Ø,16Ø,162,251,154,76              <Ø49>
2 DATA 116,196,165,43,164,44,32,8,196,169,
  54,16Ø,16Ø,76,15,228,147,42,42              <117>
3 DATA 42,42,32,72,89,8Ø,82,65,45,83,89,83
  ,84,69,77,32,42,42,42,42,13,Ø               <123>
4 DATA Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,169
  ,148,16Ø,16Ø,141,48,3,14Ø,49,3              <Ø21>
5 DATA 169,152,16Ø,164,141,5Ø,3,14Ø,51,3,9
  6,44,17,145,32,52,247,32,225,255            <217>
6 DATA 2Ø8,9,32,82,253,32,95,16Ø,76,213,25
  4,76,86,255,76,2Ø2,245,76,135               <121>
7 DATA 247,76,144,245,133,147,169,Ø,133,14
  4,165,186,2Ø8,3,76,15Ø,247,2Ø1              <Ø1Ø>
8 DATA 3,24Ø,249,144,228,164,183,2Ø8,3,76,
  147,247,32,188,228,169,96,133               <Ø14>
9 DATA 185,32,149,244,165,186,32,2Ø,238,16
  5,185,32,2Ø6,238,32,25,239,133              <Ø17>
1Ø DATA 174,165,144,74,74,176,193,32,25,23
   9,133,175,32,193,228,169,253,37            <Ø99>
11 DATA 144,133,144,165,147,2Ø8,178,16Ø,Ø,
   177,187,2Ø1,36,24Ø,17Ø,169,1,133           <254>
12 DATA 167,169,Ø,133,144,165,167,32,23,23
   8,169,111,32,192,238,165,144,16            <Ø72>
13 DATA 11,23Ø,167,165,167,2Ø1,16,2Ø8,23Ø,
   76,59,161,165,167,2Ø1,8,24Ø,239            <Ø88>
14 DATA 16Ø,Ø,185,35,161,24Ø,6,32,21Ø,255,
   2ØØ,2Ø8,245,32,225,255,2Ø8,251             <181>
15 DATA 76,59,161,13,78,85,82,32,7Ø,76,79,
   8Ø,8Ø,89,32,65,78,83,67,72,65,76           <181>
16 DATA 84,69,78,33,Ø,169,162,162,162,133,
   167,134,168,169,Ø,162,3,133,169            <Ø49>
17 DATA 134,17Ø,169,8,32,23,238,169,111,32
   ,192,238,169,77,32,228,238,169             <177>
18 DATA 45,32,228,238,169,87,32,228,238,16
   Ø,Ø,165,169,32,228,238,165,17Ø             <2Ø4>
19 DATA 32,228,238,169,3Ø,32,228,238,177,1
   67,32,228,238,2ØØ,192,3Ø,144,246           <Ø13>
2Ø DATA 32,4,239,24,165,167,1Ø5,3Ø,133,167
   ,144,3,23Ø,168,24,165,169,166,17Ø          <Ø1Ø>
21 DATA 1Ø5,3Ø,133,169,144,2,23Ø,17Ø,224,5
   ,144,173,2Ø1,Ø,144,169,169,8,32            <Ø13>
22 DATA 23,238,169,111,32,192,238,169,77,3
   2,228,238,169,45,32,228,238,169            <Ø24>
23 DATA 69,32,228,238,169,139,32,228,238,1
   69,4,32,228,238,32,1,239,32,2Ø9            <Ø57>
24 DATA 161,166,174,164,175,88,96,12Ø,169,
   1,133,167,16Ø,255,32,69,162,192            <194>
25 DATA 255,24Ø,55,162,2,165,167,24Ø,2,162
   ,4,173,Ø,175,2Ø8,7,238,1,175,173           <24Ø>
26 DATA 1,175,44,169,Ø,133,168,189,Ø,175,1
   45,174,23Ø,174,2Ø8,2,23Ø,175,232           <Ø4Ø>
27 DATA 228,168,2Ø8,24Ø,162,Ø,134,167,173,
   Ø,175,2Ø8,198,169,64,133,144,24            <2Ø3>
28 DATA 96,169,29,56,96,169,128,141,31,145
   ,173,31,145,41,2,24Ø,249,169,Ø             <Ø7Ø>
29 DATA 141,31,145,162,6,2Ø2,2Ø8,253,162,4
   ,173,31,145,1Ø6,1Ø2,176,1Ø6,1Ø2            <243>
3Ø DATA 176,234,234,2Ø2,2Ø8,242,165,176,73
   ,255,96,32,26,162,2Ø1,255,24Ø,248          <212>
31 DATA 16Ø,Ø,169,128,141,31,145,173,31,14
   5,41,2,24Ø,249,169,Ø,141,31,145            <128>
32 DATA 162,7,2Ø2,2Ø8,253,173,31,145,1Ø6,1
   Ø2,176,1Ø6,1Ø2,176,234,234,234             <148>
33 DATA 234,234,173,31,145,1Ø6,1Ø2,176,1Ø6
   ,1Ø2,176,234,234,234,234,234,173           <157>
34 DATA 31,145,1Ø6,1Ø2,176,1Ø6,1Ø2,176,234
   ,234,234,234,234,173,31,145,1Ø6            <Ø9Ø>
35 DATA 1Ø2,176,1Ø6,1Ø2,176,165,176,73,255
   ,153,Ø,175,2ØØ,2Ø8,173,96,165,Ø            <17Ø>
36 DATA 41,6,2Ø1,2,24Ø,3,76,158,253,234,16
   9,5,133,9,162,9Ø,134,75,162,Ø,169          <Ø87>
37 DATA 82,133,36,32,86,245,8Ø,254,184,173
   ,1,28,197,36,24Ø,9,198,75,2Ø8,239          <2Ø3>
38 DATA 169,1Ø,76,1Ø5,249,8Ø,254,184,173,1
   ,28,149,37,232,224,7,2Ø8,243,32            <164>
39 DATA 151,244,165,22,69,23,69,24,69,25,6
   9,26,24Ø,7,198,9,2Ø8,192,76,3Ø             <192>
4Ø DATA 244,165,24,197,6,24Ø,3,76,11,244,1
   33,34,169,6,133,49,76,6Ø,4,165             <Ø34>
41 DATA 18,166,19,133,22,134,23,165,6,133,
   24,165,7,133,25,169,Ø,69,22,69             <236>
42 DATA 23,69,24,69,25,133,26,32,52,249,16
   2,9Ø,32,86,245,16Ø,Ø,8Ø,254,184            <139>
43 DATA 173,1,28,217,36,Ø,24Ø,6,2Ø2,2Ø8,23
   7,76,81,245,2ØØ,192,8,2Ø8,234,32           <246>
44 DATA 86,245,8Ø,254,184,173,1,28,145,48,
   2ØØ,2Ø8,245,16Ø,186,8Ø,254,184             <156>
```

```
45 DATA 173,1,28,153,Ø,1,2ØØ,2Ø8,244,32,22
   4,248,165,56,197,71,24Ø,3,76,246          <244>
46 DATA 244,32,233,245,197,58,24Ø,3,76,2,2
   45,16Ø,Ø,169,85,32,82,4,185,Ø,6           <1Ø7>
47 DATA 133,119,44,Ø,24,16,251,169,16,141,
   Ø,24,44,Ø,24,48,251,162,Ø,138,1Ø2         <139>
48 DATA 119,42,42,1Ø2,119,42,42,141,Ø,24,1
   38,1Ø2,119,42,42,1Ø2,119,42,42            <155>
49 DATA 141,Ø,24,138,1Ø2,119,42,42,1Ø2,119
   ,42,42,141,Ø,24,138,1Ø2,119,42            <151>
5Ø DATA 42,1Ø2,119,42,42,141,Ø,24,162,2,2Ø
   2,2Ø8,253,169,15,141,Ø,24,2ØØ,2Ø8         <Ø37>
51 DATA 173,234,234,234,234,234,234,234,17
   3,Ø,28,9,8,141,Ø,28,173,Ø,6,2Ø8           <191>
52 DATA 3,76,158,253,197,24,2Ø8,249,133,6,
   173,1,6,133,7,76,1Ø1,3,133,119            <163>
53 DATA 44,Ø,24,16,251,169,16,141,Ø,24,44,
   Ø,24,48,251,162,4,169,Ø,1Ø2,119           <213>
54 DATA 42,42,1Ø2,119,42,42,141,Ø,24,2Ø2,2
   Ø8,24Ø,234,234,234,234,234,234            <254>
55 DATA 169,15,141,Ø,24,96,96,133,Ø,88,165
   ,Ø,48,252,12Ø,96,12Ø,234,234,234          <1Ø7>
56 DATA 234,234,234,165,24,141,Ø,6,133,6,1
   65,25,141,1,6,133,7,169,4,133,12Ø         <12Ø>
57 DATA 169,226,32,13Ø,4,2Ø1,2,144,51,16Ø,
   Ø,132,12Ø,164,12Ø,185,219,254,24Ø         <2Ø3>
58 DATA 18,88,32,118,214,12Ø,169,226,32,13
   Ø,4,2Ø1,2,144,26,23Ø,12Ø,2Ø8,231          <242>
59 DATA 169,192,32,13Ø,4,169,226,32,13Ø,4,
   2Ø1,2,144,8,169,255,32,82,4,76            <213>
6Ø DATA 34,235,173,Ø,6,24Ø,248,197,24,24Ø,
   196,173,Ø,6,133,6,173,1,6,133,7           <216>
61 DATA 76,16Ø,4,165,186,2Ø1,4,176,3,76,13
   3,246,12Ø,186,142,216,168,173,25          <Ø92>
62 DATA 3,141,217,168,173,24,3,141,218,168
   ,169,11,141,24,3,169,166,141,25           <247>
63 DATA 3,173,44,145,141,31,1,88,169,97,13
   3,185,162,44,16Ø,Ø,177,187,2Ø1            <1Ø4>
64 DATA 42,2Ø8,1Ø,162,76,198,183,23Ø,187,2
   Ø8,2,23Ø,188,142,46,168,169,165           <Ø9Ø>
65 DATA 72,169,24Ø,72,165,183,2Ø8,3,76,132
   ,247,32,149,244,32,4Ø,247,165,186         <Ø64>
66 DATA 32,23,238,165,185,32,192,238,169,Ø
   ,133,144,32,228,238,32,228,238            <241>
67 DATA 32,4,239,165,144,24Ø,2,24,96,169,1
   59,162,166,133,172,134,173,169            <166>
68 DATA 7Ø,162,1,133,2Ø,134,21,162,5,165,1
   86,32,23,238,169,111,32,192,238           <Ø76>
69 DATA 16Ø,253,185,154,165,32,228,238,2ØØ
   ,2Ø8,247,165,2Ø,32,228,238,165            <142>
7Ø DATA 21,32,228,238,169,31,32,228,238,16
   Ø,Ø,177,172,32,228,238,2ØØ,192            <122>
71 DATA 31,144,246,32,4,239,165,172,24,1Ø5
   ,31,133,172,144,2,23Ø,173,165,2Ø          <Ø91>
72 DATA 24,1Ø5,31,133,2Ø,144,2,23Ø,21,2Ø2,
   2Ø8,18Ø,165,186,32,23,238,169,111         <Ø14>
73 DATA 32,192,238,16Ø,251,185,159,165,32,
   228,238,2ØØ,2Ø8,247,32,4,239,12Ø          <129>
74 DATA 162,Ø,142,31,145,169,22Ø,141,44,14
   5,189,55,167,157,Ø,175,232,2Ø8            <Ø13>
75 DATA 247,173,31,145,41,2,2Ø8,249,32,131
   ,166,162,Ø,189,55,168,157,Ø,175           <Ø3Ø>
76 DATA 232,2Ø8,247,32,131,166,32,21Ø,251,
   162,1,165,172,157,1,175,232,165           <Ø32>
77 DATA 173,157,1,175,232,32,17,253,176,23
   ,16Ø,Ø,177,172,157,1,175,232,32           <Ø84>
78 DATA 27,253,224,255,144,236,32,17,253,1
   76,3,169,255,44,169,Ø,72,141,Ø            <112>
79 DATA 175,142,1,175,32,131,166,162,1,1Ø4
   ,2Ø8,211,24,169,Ø,72,173,31,1,141         <242>
8Ø DATA 44,145,12Ø,173,218,168,141,24,3,17
   3,217,168,141,25,3,1Ø4,174,216            <132>
81 DATA 168,154,96,56,173,31,1,9,2,141,31,
   1,176,217,185,Ø,175,133,149,162           <Ø74>
82 DATA Ø,173,31,145,41,1,24Ø,249,173,31,1
   45,41,2,24Ø,5,232,2Ø8,246,24Ø,19Ø         <244>
83 DATA 169,222,141,44,145,42,42,42,6,149,
   42,42,42,42,6,149,42,42,141,44            <Ø84>
84 DATA 145,42,42,42,6,149,42,42,42,6,1
   49,42,42,141,44,145,42,42,42,6            <119>
85 DATA 149,42,42,42,42,6,149,42,42,141,44
   ,145,42,42,42,6,149,42,42,42,42           <144>
86 DATA 6,149,42,42,141,44,145,234,234,234
   ,234,234,169,22Ø,141,44,145,2ØØ           <167>
87 DATA 2Ø8,148,96,16Ø,Ø,152,89,Ø,175,2ØØ,
   2Ø8,25Ø,133,2Ø,32,22,166,136,165          <224>
88 DATA 2Ø,76,25,166,77,45,87,77,45,69,187
   ,1,16Ø,Ø,132,17,169,2,141,Ø,24            <1Ø9>
89 DATA 169,4,44,Ø,24,24Ø,251,169,Ø,141,Ø,
   24,162,4,2Ø2,2Ø8,253,162,1Ø,173           <Ø46>
9Ø DATA Ø,24,74,38,133,74,74,38,133,97,Ø,1
   Ø1,Ø,173,Ø,24,74,38,133,74,74,38          <19Ø>
91 DATA 133,97,Ø,1Ø1,Ø,173,Ø,24,74,38,133,
   74,74,38,133,97,Ø,1Ø1,Ø,173,Ø,24          <Ø54>
92 DATA 142,Ø,24,74,38,133,74,74,38,133,16
   5,133,145,48,69,17,133,17,2ØØ,2Ø8         <Ø71>
93 DATA 166,136,96,133,49,32,7Ø,1,177,48,7
   2,32,74,1,1Ø4,145,48,165,17,24Ø          <251>
94 DATA 237,76,67,232,12Ø,169,1Ø,141,Ø,24,
   162,Ø,136,2Ø8,253,2Ø2,2Ø8,25Ø,169        <212>
95 DATA 1Ø,133,1Ø5,169,Ø,133,48,169,3,32,1
   66,1,169,4,32,166,1,76,61,4,165          <Ø38>
96 DATA Ø,162,1,134,Ø,41,2,24Ø,16,166,152,
   134,5Ø,32,46,4,32,175,3,32,3,4           <18Ø>
97 DATA 76,1Ø5,249,162,8,32,53,3,162,1Ø,32
   ,53,3,162,8,32,245,3,162,1Ø,32           <Ø53>
98 DATA 245,3,165,14Ø,48,232,76,1Ø5,249,13
   4,5Ø,134,152,165,14Ø,16,97,181           <142>
99 DATA 131,2Ø8,93,189,49,4,32,166,1,2ØØ,1
   66,5Ø,169,128,149,131,165,128,149        <224>
1ØØ DATA Ø,165,129,149,1,177,48,24Ø,28,32,
    33,241,166,13Ø,246,181,2Ø8,2,246        <2Ø5>
1Ø1 DATA 187,16Ø,Ø,165,128,145,48,2ØØ,165,
    129,145,48,165,128,197,34,24Ø,2         <Ø66>
1Ø2 DATA 132,14Ø,166,5Ø,169,Ø,133,48,133,5
    1,133,46,133,54,133,12,133,8Ø,169       <164>
1Ø3 DATA 187,133,52,189,49,4,133,47,32,233
    ,245,133,58,189,5Ø,4,32,163,247         <Ø91>
1Ø4 DATA 166,5Ø,181,131,24Ø,82,32,46,4,173
    ,Ø,28,41,16,2Ø8,3,76,129,245,32         <223>
1Ø5 DATA 16,245,162,9,8Ø,254,184,2Ø2,2Ø8,2
    5Ø,169,255,141,3,28,173,12,28,41        <222>
1Ø6 DATA 31,Ø,192,141,12,28,169,255,162,5,
    141,1,28,184,8Ø,254,184,2Ø2,2Ø8         <238>
1Ø7 DATA 25Ø,16Ø,187,177,12,8Ø,254,184,141
    ,1,28,22Ø,2Ø8,245,177,48,8Ø,254         <2Ø1>
1Ø8 DATA 184,141,1,28,2ØØ,2Ø8,245,8Ø,254,7
    6,Ø,254,98,134,5Ø,76,36,4,134,152       <247>
1Ø9 DATA 181,131,24Ø,244,32,46,4,32,1Ø,245
    ,16Ø,187,177,12,8Ø,254,184,77,1         <122>
11Ø DATA 28,2Ø8,25,2ØØ,2Ø8,243,177,48,8Ø,2
    54,184,77,1,28,2Ø8,12,2ØØ,192,253       <237>
111 DATA 225,241,166,5Ø,169,Ø,149,131,96,7
    6,197,246,189,49,4,133,49,189,5Ø        <Ø84>
112 DATA 4,133,13,96,5,1,6,4,88,32,25,241,
    169,132,213,167,24Ø,5,149,167,32        <196>
113 DATA 66,2Ø8,169,64,141,249,2,169,1,133
    ,131,32,7,2Ø9,144,3,76,248,2Ø7,32       <16Ø>
114 DATA 82,222,246,181,169,Ø,133,139,133,
    141,169,128,133,14Ø,165,128,133         <233>
115 DATA 6,169,224,133,Ø,165,Ø,48,252,24Ø,
    36,2Ø1,1,24Ø,234,165,24,133,6,165       <ØØ8>
116 DATA 25,133,7,162,Ø,169,176,32,125,213
    ,32,153,213,169,226,32,125,213,32       <Ø96>
117 DATA 153,213,165,14Ø,2Ø8,2Ø4,24Ø,2Ø6,7
    6,35,219,Ø                              <23Ø>
19Ø OPEN 1,8,15:CLOSE 1:IF ST=Ø THEN 2ØØ    <Ø65>
195 PRINT"BITTE FLOPPY ANSCHLIESSEN, DANN
    <SPACE>"                                <183>
196 GET T$:IF T$<>" "THEN 196               <Ø23>
197 GOTO 19Ø                                <Ø13>
2ØØ OPEN 8,8,8,"HYPRA-SYST VC2Ø,P,W"        <Ø68>
21Ø PRINT#8,CHR$(Ø)CHR$(16Ø);               <ØØ2>
22Ø FOR I=Ø TO 2264:READ A:P=P+A:PRINT#8,C
    HR$(A);:NEXT                            <Ø19>
23Ø CLOSE 8                                 <Ø41>
24Ø IF P<>266756 THEN PRINT"DATAS NICHT OK
    ":STOP                                  <166>
25Ø PRINT"DATAS OK"                         <193>
```

Listing. »Hypra-System«. Beim Starten wird das eigentliche Programm auf der Diskette generiert. Beachten Sie die Eingabehinweise auf Seite 129.

SMON für den VC 20

Der Super-Maschinensprache-Monitor auf dem VC 20. Mit mindestens 16 KByte können Sie die Vorteile des SMON nutzen. Für den Assembler-Programmierer ist SMON für den VC 20 ein unentbehrliches Werkzeug.

SMON VC 20 ist eine Umsetzung des bekannten SMON für den C64, veröffentlicht in Sonderheft 8/85, auf dem VC 20. Die Stärken des SMON sind vor allem seine mächtigen Suchbefehle. Als Zusatz enthält er einen Disketten-Monitor und natürlich einen Disassembler. Der Wermutstropfen beim SMON VC 20 ist, daß man die Trace-Befehle nicht korrekt umsetzen kann. Das liegt am etwas höheren Systemtakt des VC 20 und den damit verbundenen Timing-Problemen.
Sämtliche Standard-Befehle für 6502-Assembler sind natürlich im SMON vorhanden. Dazu kommen Umrechnungsroutinen für Hexadezimal-Zahlen, komfortable Verschiebe- und Suchbefehle. Der Direktassembler des SMON verarbeitet sogar Label.
»SMON VC 20« liegt beim VC 20 im Bereich ab $A000. Tippen sie das Listing »SMON VC 20« ein, speichern Sie es und starten es dann. Daraufhin wird auf der Floppy das Programm »SMON.EX VC 20« generiert. Dieses laden Sie dann mit »,8,1« und starten es mit SYS 40960. Die Übersicht aller zur Verfügung stehenden Befehle entnehmen Sie bitte der Tabelle.
Für Datasette: Da das Ladeprogramm »SMON VC 20« auf die Diskette ein File schreibt, müssen Datasetten-Besitzer einen anderen Weg gehen. Tippen Sie im Generator-Programm die Zeilen ab Zeilennummer 300 nicht mit ein. Statt dessen tippen Sie folgende Zeilen ein:

```
300 FOR I=0 TO 4090:READ A:POKE 40960+I,A:P=P+A:
NEXT
310 IF P<>523777 THEN PRINT "DATAS NICHT OK":STOP
320 PRINT "DATAS OK"
RUN
```

Wenn die Meldung »Datas ok« erscheint, geben Sie ein:

```
POKE 43,0:POKE 44,160:POKE 45,0:POKE 46,176
<RETURN>
SAVE "SMON.EX VC20" <RETURN>
```

Damit steht SMON auf Datasette zur Verfügung.

(K.H. Templin/og)

Befehlsübersicht zum SMON	
	Alle Eingaben erfolgen in der hexadezimalen Schreibweise. In Klammern angegebene Adreßeingaben können entfallen. SMON benutzt dann sinnvolle, vorgegebene Werte. Bei allen Ausgabe-Befehlen ist gleichzeitig die Ausgabe auf einem Drucker möglich. Dazu werden die Befehle geSHIFTet eingegeben.
A	**4000** (Assembler) symbolischer Assembler (Verarbeitung von Label möglich. Label bestehen aus dem Buchstaben »M« und einer zweistelligen Hex-Zahl von 01 bis 30.) Startadresse $4000
B	**4000 4200** (Basic-Data) erzeugt Basic-DATA-Zeilen aus Maschinenprogramm im Bereich von $4000 bis $41FF
C	**4010 4200 4013 4000 4200** (Convert) in ein Programm, das von $4000 bis $4200 im Speicher steht, soll bei 4010 ein 3-Byte-Befehl eingefügt werden. Dazu wird das Programm ab $4010 bis 4200 auf die neue Adresse $4013 verschoben. Alle absoluten Adressen, die innerhalb des Programmbereichs ($4000 bis $4200) stehen, werden umgerechnet, so daß die Sprungziele stimmen.
D	**4000 (4100)** (Disassembler) disassembliert den Bereich von $4000 (bis $4100) mit Ausgabe der Hex-Werte. Änderungen sind durch Überschreiben der Befehle möglich.
F	(Find) findet Zeichenketten (F), absolute Adressen (FA), relative Sprünge (FR), Tabellen (FT), Zeropage-Adressen (FZ) und Immediate-Befehle (FI)
G	**(4000)** (Go) startet ein Maschinenprogramm, das bei $4000 im Speicher beginnt
I	**01** (I/O-Gerät) stellt die Gerätenummer für Floppy (08 oder 09) oder Datasette (01) ein
K	**A000 (A500)** (Kontrolle) zum schnellen Durchsuchen des Bereichs von $A000 (bis $A500) nach ASCII-Zeichen (32 Byte pro Zeile). Änderungen sind durch Überschreiben der ASCII-Zeichen möglich.
L	**(4000)** (Load) lädt ein Maschinenprogramm an die richtige oder eine angegebene Adresse ($4000)
M	**4000 (4400)** (Memory Dump) gibt den Inhalt des Speichers von $4000 (bis $43FF) in Hex-Byte und ASCII-Code aus. Änderungen sind durch Überschreiben der Hex-Zahlen möglich.
O	**4000 4500 AA** (Occupy) füllt den Speicherbereich von $4000 bis $4500 mit vorgegebenem Byte ($AA) aus
P	**05** (Printer) setzt Geräteadresse 5 für Drucker
R	(Register) zeigt die Registerinhalte und Flags an. Änderungen sind durch Überschreiben möglich.
S	**"Test" 4000 5000** (Save) speichert ein Programm von $4000 bis $4FFF unter dem Namen »Test« ab
V	**6000 6200 4000 4100 4200** (Verschieben) ändert in einem Programm von $4100 bis $41FF alle absoluten Adressen, die sich auf den Bereich von $6000 bis $6200 beziehen, auf einen neuen Bereich, der bei $4000 beginnt.
W	**4000 4300 5000** (Write) verschiebt den Speicherinhalt von $4000 bis $42FF nach $5000 ohne Umrechnung der Adressen (zum Beispiel Tabellen)
X	(Exit) springt aus dem Monitor-Programm ins Basic zurück

49152
Dezimalzahl umrechnen

$ 002B
4stellige Hex-Zahl umrechnen

% 01101010
8stellige Binärzahl umrechnen

? 0344 + 5234
Addition oder Subtraktion zweier 4stelliger Hex-Zahlen

= 4000 5000 (Vergleich)
vergleicht den Speicherinhalt ab $4000 mit dem ab $5000

Z (Diskmonitor)
ruft den Diskmonitor auf. Dieser verfügt über folgende Befehle:

R (12 01) (Read) (Nur im Diskmonitor)
liest Track $12, Sektor $01 von der Diskette in einen Puffer im Speicher. Fehlt die Angabe von Track und Sektor, wird der logisch (!) nächste Sektor gelesen.

W (12 01) (Write) (Nur im Diskmonitor)
schreibt den Puffer im Speicher nach Track $12, Sektor $01 auf die Diskette. Ohne Angabe von Track und Sektor werden die letzten Eingaben von »R« benutzt.

M (Memory Dump) (Nur im Diskmonitor)
zeigt den Pufferinhalt als Hexdump (wie normales »M«). Weitere Ausgabe mit CBM-Taste, Abbruch mit STOP. Werte können durch Überschreiben geändert werden.

X (Exit) (Nur im Diskmonitor)
springt in SMON zurück

F (weitere Disketten-Befehle initialisieren) (Nur im Diskmonitor) sind die Befehle initialisiert, gilt:

M (07)
Memory Dump (Floppy-RAM/ROM)
V 6000 0400
Verschieben eines 256-Byte-Blocks von $6000 in den Laufwerkspuffer 1 beziehungsweise in das Floppy-RAM
@ normale Disketten-Befehle senden
X zurück zum normalen Disketten-Monitor

```
Ø DATA 169,2Ø,141,22,3,169,162,141,23,3,Ø,
  39,35,36,37,44,58,59,61,63,65              <145>
1 DATA 66,67,68,7Ø,71,73,75,76,77,79,8Ø,82
  ,83,84,86,87,88,9Ø,Ø,Ø,Ø,Ø,218            <165>
2 DATA 17Ø,45,169,7,169,27,169,251,166,28,
  164,181,163,244,17Ø,153,168,2Ø8            <Ø6Ø>
3 DATA 166,1Ø7,169,6Ø,17Ø,92,165,16,171,22
  6,163,67,168,182,17Ø,77,168,248            <Ø39>
4 DATA 163,192,169,6Ø,168,133,163,77,168,2
  4Ø,171,66,17Ø,21Ø,169,1Ø9,163              <252>
5 DATA 8,174,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,255,255,1,Ø,6
  5,9Ø,73,82,84,128,32,64,16,Ø,2             <184>
6 DATA 1,1,2,Ø,145,145,13,83,217,49,55,5Ø,
  13,Ø,125,76,125,169,13,13,32,32            <Ø95>
7 DATA 8Ø,67,32,32,83,82,32,65,67,32,88,82
  ,32,89,82,32,83,8Ø,Ø,32,78,86              <Ø45>
8 DATA 45,66,68,73,9Ø,67,Ø,2,4,1,44,Ø,44,8
  9,41,88,157,31,255,28,28,31,31             <Ø56>
9 DATA 31,28,223,28,31,223,255,255,3,31,12
  8,9,32,12,4,16,1,17,2Ø,15Ø,28              <218>
1Ø DATA 25,148,19Ø,1Ø8,3,19,1,2,2,3,3,2;2,
   2,2,2,2,3,3,2,3,3,3,2,Ø,64,64,128         <211>
11 DATA 128,32,16,37,38,33,34,129,13Ø,33,1
   3Ø,132,8,8,231,231,231,231,227            <247>
12 DATA 227,227,227,227,227,227,227,227,22
   7,231,167,231,231,243,243,247,223         <178>
13 DATA 38,7Ø,6,1Ø2,65,129,225,1,16Ø,162,1
   61,193,33,97,132,134,23Ø,198,224          <Ø64>
14 DATA 192,36,76,32,144,176,24Ø,48,2Ø8,16
   ,8Ø,112,12Ø,Ø,24,216,88,184,2Ø2           <197>
15 DATA 136,232,2ØØ,234,72,8,1Ø4,4Ø,64,96,
   17Ø,168,186,138,154,152,56,248            <2Ø9>
16 DATA 137,156,158,178,42,74,1Ø,1Ø6,79,35
   ,147,179,243,51,211,19,83,115,82          <136>
17 DATA 76,65,82,69,83,83,79,76,76,76,67,6
   5,65,83,83,73,68,67,67,66,74,74           <217>
18 DATA 66,66,66,66,66,66,66,66,83,66,67,6
   7,67,67,68,68,73,73,78,8Ø,8Ø,8Ø           <231>
19 DATA 8Ø,82,82,84,84,84,84,84,84,83,83,7
   9,83,83,79,79,84,66,82,68,68,68           <161>
2Ø DATA 77,78,68,84,84,78,69,8Ø,8Ø,73,77,8
   3,67,67,69,77,78,8Ø,86,86,69,82           <Ø69>
21 DATA 76,76,76,76,69,69,78,78,79,72,72,7
   6,76,84,84,65,65,83,88,88,89,69           <199>
22 DATA 69,76,82,76,82,82,65,67,65,89,88,6
   5,8Ø,68,67,89,88,67,67,88,89,84           <227>
23 DATA 8Ø,82,67,83,81,73,69,76,67,83,73,7
   5,67,68,73,86,88,89,88,89,8Ø,65           <Ø61>
24 DATA 8Ø,65,8Ø,73,83,88,89,88,65,83,65,6
   7,68,8,132,129,34,33,38,32,128            <234>
25 DATA 3,32,28,2Ø,2Ø,16,4,12,216,169,8,14
   1,176,2,169,4,141,175,2,234,234           <Ø31>
26 DATA 234,234,234,234,234,234,234,234,23
   4,234,234,162,5,1Ø4,157,168,2,2Ø2         <121>
27 DATA 16,249,173,169,2,2Ø8,3,2Ø6,168,2,2
   Ø6,169,2,186,142,174,2,169,82,76          <234>
28 DATA 255,162,32,194,162,24Ø,11,32,126,1
   62,141,169,2,165,252,141,168,2            <Ø63>
29 DATA 96,162,164,32,128,162,32,128,162,2
   Ø8,28,32,126,162,169,254,133,253          <199>
3Ø DATA 169,255,133,254,32,194,162,2Ø8,12,
   141,119,2,23Ø,198,96,32,126,162           <242>
31 DATA 44,162,251,32,141,162,149,1,32,154
   ,162,149,Ø,232,232,96,32,2Ø2,162          <Ø52>
32 DATA 2Ø1,32,24Ø,249,2Ø1,44,24Ø,245,2Ø8,
   3,32,2Ø2,162,32,175,162,1Ø,1Ø,1Ø          <175>
33 DATA 1Ø,133,18Ø,32,2Ø2,162,32,175,162,5
   ,18Ø,96,2Ø1,58,144,2,1Ø5,8,41,15          <119>
34 DATA 96,32,2Ø2,162,2Ø1,32,24Ø,249,198,2
   11,96,32,2Ø7,255,198,211,2Ø1,13           <Ø44>
35 DATA 96,32,2Ø7,255,2Ø1,13,2Ø8,248,169,6
   3,32,21Ø,255,174,174,2,154,162            <137>
36 DATA Ø,134,198,32,81,163,161,2Ø9,2Ø1,39
   ,24Ø,17,2Ø1,58,24Ø,13,2Ø1,59,24Ø          <115>
37 DATA 9,2Ø1,44,24Ø,5,169,46,32,21Ø,255,3
   2,2Ø2,162,2Ø1,46,24Ø,249,133,172          <Ø5Ø>
38 DATA 41,127,162,32,221,1Ø,16Ø,24Ø,5,2Ø2
   ,2Ø8,248,24Ø,194,32,21,163,76,214         <Ø79>
39 DATA 162,138,1Ø,17Ø,232,189,41,16Ø,72,2
   Ø2,189,41,16Ø,72,96,165,252,32            <Ø44>
4Ø DATA 42,163,165,251,72,74,74,74,74,32,5
   3,163,1Ø4,41,15,2Ø1,1Ø,144,2,1Ø5          <Ø37>
41 DATA 6,1Ø5,48,76,21Ø,255,169,13,32,21Ø,
   255,138,76,21Ø,255,32,76,163,169          <129>
42 DATA 32,76,21Ø,255,169,13,76,21Ø,255,13
   3,187,132,188,16Ø,Ø,177,187,24Ø           <Ø8Ø>
43 DATA 6,32,21Ø,255,2ØØ,2Ø8,246,96,23Ø,25
   1,2Ø8,2,23Ø,252,96,169,6,141,134          <253>
44 DATA 2,169,27,141,15,144,234,234,234,23
   4,234,234,234,174,174,2,154,76            <161>
45 DATA 116,196,16Ø,16Ø,169,14Ø,32,86,163,
   162,59,32,64,163,173,168,2,133            <136>
46 DATA 252,173,169,2,133,251,32,35,163,32
   ,76,163,162,251,189,175,1,32,42           <175>
47 DATA 163,32,76,163,232,2Ø8,244,96,234,2
   34,234,234,234,32,78,162,162,251          <246>
48 DATA 32,2Ø2,162,32,154,162,157,175,1,23
   2,2Ø8,244,96,234,234,234,234,234          <Ø77>
49 DATA 234,234,234,133,17Ø,169,32,16Ø,9,3
   2,21Ø,255,6,17Ø,169,48,1Ø5,Ø,136          <195>
5Ø DATA 2Ø8,244,96,32,73,162,174,174,2,154
   ,162,25Ø,189,174,1,72,232,2Ø8,249         <Ø28>
51 DATA 1Ø4,168,1Ø4,17Ø,1Ø4,64,32,1ØØ,162,
   162,58,32,64,163,32,35,163,16Ø            <237>
```

Listing. »SMON VC 20« benötigt mindestens eine 16-KByte-Erweiterung

```
52 DATA 18,162,Ø,32,76,163,161,251,32,42,1
   63,161,251,32,57,164,208,241,32          <186>
53 DATA 93,164,144,224,96,32,126,162,16Ø,1
   8,162,Ø,32,2Ø2,162,32,154,162,129        <1Ø8>
54 DATA 251,193,251,24Ø,3,76,2Ø9,162,32,57
   ,164,2Ø8,236,96,2Ø1,32,144,12,2Ø1        <126>
55 DATA 96,144,1Ø,2Ø1,192,144,4,2Ø1,219,14
   4,4,169,46,41,63,41,127,145,2Ø9          <252>
56 DATA 173,134,2,145,243,32,1Ø3,163,2ØØ,1
   92,22,96,32,111,164,76,1Ø2,164           <194>
57 DATA 32,1Ø3,163,165,251,197,253,165,252
   ,229,254,96,32,148,164,32,134,164        <Ø69>
58 DATA 24Ø,14,32,134,164,24Ø,251,2Ø1,32,2
   Ø8,5,141,119,2,23Ø,198,96,32,228         <1Ø7>
59 DATA 255,72,32,225,255,24Ø,2,1Ø4,96,76,
   214,162,16Ø,22,36,172,16,246,132         <147>
6Ø DATA 2ØØ,132,2Ø8,169,255,32,195,255,169
   ,255,133,184,133,185,173,175,2           <248>
61 DATA 133,186,32,192,255,162,Ø,134,211,2
   Ø2,32,2Ø1,255,32,2Ø7,255,32,21Ø          <125>
62 DATA 255,2Ø1,13,2Ø8,246,32,2Ø4,255,169,
   145,76,21Ø,255,16Ø,0,177,251,36          <184>
63 DATA 17Ø,48,2,8Ø,12,162,31,221,6Ø,161,2
   4Ø,47,2Ø2,224,21,2Ø8,246,162,4           <133>
64 DATA 221,73,161,24Ø,33,221,77,161,24Ø,3
   Ø,2Ø2,2Ø8,243,162,56,221,17,161          <2ØØ>
65 DATA 24Ø,2Ø,2Ø2,224,22,2Ø8,246,177,251,
   61,251,16Ø,93,17,161,24Ø,5,2Ø2           <226>
66 DATA 2Ø8,243,162,Ø,134,173,138,24Ø,15,1
   62,17,177,251,61,181,16Ø,93,198          <1Ø6>
67 DATA 16Ø,24Ø,3,2Ø2,2Ø8,243,189,234,16Ø,
   133,171,189,216,16Ø,133,182,166          <185>
68 DATA 173,96,16Ø,1,177,251,17Ø,2ØØ,177,2
   51,16Ø,16,196,171,2Ø8,7,32,74,165        <234>
69 DATA 16Ø,3,2Ø8,2,164,182,142,174,Ø,141,
   175,Ø,96,16Ø,1,177,251,16,1,136          <Ø48>
7Ø DATA 56,1Ø1,251,17Ø,232,24Ø,1,136,152,1
   Ø1,252,96,162,Ø,134,17Ø,32,1ØØ           <Ø97>
71 DATA 162,32,14Ø,165,165,173,2Ø1,22,24Ø,
   9,2Ø1,48,24Ø,5,2Ø1,33,2Ø8,17,234         <Ø92>
72 DATA 32,148,164,32,81,163,162,21,169,45
   ,32,21Ø,255,2Ø2,2Ø8,25Ø,32,93,164        <216>
73 DATA 144,217,96,162,44,32,64,163,32,35,
   163,32,76,163,32,117,166,32,2Ø3          <Ø12>
74 DATA 164,234,234,234,234,234,234,234,23
   4,234,234,234,234,234,234,234,234        <146>
75 DATA 234,234,234,234,234,234,234,234,23
   4,234,234,234,234,234,234,234,234        <1Ø7>
76 DATA 234,234,234,234,234,16Ø,Ø,166,173,
   2Ø8,17,162,3,169,42,32,21Ø,255           <Ø93>
77 DATA 2Ø2,2Ø8,248,36,17Ø,48,133,76,1Ø6,1
   66,36,17Ø,8Ø,41,169,8,36,171,24Ø         <123>
78 DATA 35,177,251,41,252,133,173,2ØØ,177,
   251,1Ø,168,185,6Ø,3,141,174,Ø,2ØØ        <141>
79 DATA 185,6Ø,3,141,175,Ø,32,19Ø,166,164,
   182,32,147,166,32,2Ø3,164,189,91         <25Ø>
8Ø DATA 161,32,21Ø,255,189,147,161,32,21Ø,
   255,189,2Ø3,161,32,21Ø,255,169           <113>
81 DATA 32,36,171,24Ø,3,32,73,163,162,32,1
   69,4,36,171,24Ø,2,162,4Ø,138,32          <Ø6Ø>
82 DATA 21Ø,255,36,171,8Ø,5,169,35,32,21Ø,
   255,32,44,165,136,24Ø,22,169,8           <19Ø>
83 DATA 36,171,24Ø,7,169,77,32,21Ø,255,16Ø
   ,1,185,173,Ø,32,42,163,136,2Ø8           <Ø82>
84 DATA 247,16Ø,3,185,172,16Ø,36,171,24Ø,9
   ,185,175,16Ø,19Ø,178,16Ø,32,66           <198>
85 DATA 163,136,2Ø8,237,165,182,32,1Ø3,163
   ,56,233,1,2Ø8,248,96,164,211,169         <Ø8Ø>
86 DATA 32,145,2Ø9,2ØØ,192,22,144,249,96,2
   28,171,2Ø8,4,5,173,133,173,96,185        <1Ø2>
87 DATA 173,Ø,145,251,2Ø9,251,2Ø8,4,136,16
   ,244,96,1Ø4,1Ø4,96,2Ø8,28,138,5          <14Ø>
88 DATA 171,133,171,169,4,133,181,32,2Ø7,2
   55,2Ø1,32,24Ø,13,2Ø1,36,24Ø,9,2Ø1        <243>
89 DATA 4Ø,24Ø,5,2Ø1,44,24Ø,1,96,198,181,2
   Ø8,232,96,224,24,48,14,173,174           <162>
9Ø DATA Ø,56,233,2,56,229,251,141,174,Ø,16
   Ø,64,96,32,126,162,133,253,165           <245>
91 DATA 252,133,254,32,81,163,32,228,166,4
   8,251,16,246,169,Ø,133,211,32,76         <135>
92 DATA 163,32,35,163,32,76,163,32,2Ø7,255
   ,169,1,133,211,162,128,2Ø8,5,162         <ØØ1>
93 DATA 128,142,177,2,134,17Ø,32,126,162,1
   69,22,133,2ØØ,44,177,2,16,8,234          <Ø98>
94 DATA 234,32,2Ø7,255,234,234,234,169,Ø,1
   41,177,2,32,161,166,2Ø1,7Ø,2Ø8           <236>
95 DATA 22,7Ø,17Ø,1Ø4,1Ø4,162,2,181,25Ø,72
   ,181,252,149,25Ø,1Ø4,149,252,2Ø2         <192>
96 DATA 2Ø8,243,76,1ØØ,165,2Ø1,46,2Ø8,17,3
   2,154,162,16Ø,Ø,145,251,2Ø9,251          <Ø24>
97 DATA 2Ø8,4,32,1Ø3,163,2ØØ,136,96,162,25
   3,2Ø1,77,2Ø8,25,32,154,162,16Ø           <132>
98 DATA Ø,2Ø1,63,176,239,1Ø,168,165,251,15
   3,6Ø,3,165,252,2ØØ,153,6Ø,3,32           <13Ø>
99 DATA 161,166,149,169,224,253,2Ø8,4,169,
   7,133,183,232,2Ø8,24Ø,162,56,165         <159>
1ØØ DATA 166,221,91,161,24Ø,5,2Ø2,2Ø8,246,
    2Ø2,96,165,167,221,147,161,2Ø8,244      <1ØØ>
1Ø1 DATA 165,168,221,2Ø3,161,2Ø8,237,189,1
    7,161,133,173,32,161,166,16Ø,Ø,224      <235>
1Ø2 DATA 32,16,9,2Ø1,32,2Ø8,8,189,77,161,1
    33,173,76,49,168,16Ø,8,2Ø1,77,24Ø       <168>
1Ø3 DATA 32,16Ø,64,2Ø1,35,24Ø,26,32,157,16
    2,141,174,Ø,141,175,Ø,32,161,166        <246>
1Ø4 DATA 16Ø,32,2Ø1,48,144,27,2Ø1,71,176,2
    3,16Ø,128,198,211,32,161,166,32         <132>
1Ø5 DATA 157,162,141,174,Ø,32,161,166,192,
    8,24Ø,3,32,19Ø,166,132,171,162,1        <Ø91>
1Ø6 DATA 2Ø1,88,32,154,166,162,4,2Ø1,41,32
    ,154,166,162,2,2Ø1,89,32,154,166        <172>
1Ø7 DATA 165,173,41,13,24Ø,1Ø,162,64,169,8
    ,32,129,166,169,24,44,169,28,162        <247>
1Ø8 DATA 13Ø,32,129,166,16Ø,8,165,173,2Ø1,
    32,24Ø,9,19Ø,3,162,185,11,162,32        <Ø33>
1Ø9 DATA 129,166,136,2Ø8,244,165,171,16,1,
    2ØØ,2ØØ,32,138,166,198,183,165,183      <24Ø>
11Ø DATA 133,211,76,151,165,32,141,162,141
    ,175,2,96,32,141,162,141,176,2,96       <Ø96>
111 DATA 76,2Ø9,162,16Ø,2,132,188,136,132,
    185,132,187,136,132,183,32,2Ø2,162      <1Ø3>
112 DATA 2Ø1,34,2Ø8,234,32,2Ø2,162,145,187
    ,2ØØ,23Ø,183,2Ø1,34,2Ø8,244,198         <127>
113 DATA 183,173,176,2,133,186,165,172,2Ø1
    ,83,24Ø,19,32,194,162,24Ø,9,162         <Ø79>
114 DATA 195,32,128,162,169,Ø,133,185,169,
    Ø,1Ø8,48,3,162,193,32,128,162,162       <Ø72>
115 DATA 174,32,128,162,1Ø8,5Ø,3,32,126,16
    2,32,2Ø2,162,73,2,74,74,8,32,128        <Ø61>
116 DATA 162,32,81,163,4Ø,176,12,165,253,1
    Ø1,251,17Ø,165,254,1Ø1,252,56,176       <142>
117 DATA 9,165,251,229,253,17Ø,165,252,229
    ,254,168,138,132,252,133,251,132        <1Ø6>
118 DATA 98,133,99,8,169,Ø,133,211,32,117,
    166,165,252,2Ø8,15,32,73,163,165        <128>
119 DATA 251,32,42,163,165,251,32,2Ø8,163,
    24Ø,3,32,35,163,32,76,163,162,144       <117>
12Ø DATA 234,234,234,234,234,234,234,234,2
    34,4Ø,32,73,22Ø,32,221,221,234,234      <115>
121 DATA 234,234,234,76,86,163,32,141,162,
    17Ø,164,211,177,2Ø9,73,32,24Ø,163       <Ø58>
122 DATA 138,168,32,154,162,56,176,169,32,
    184,162,16Ø,8,72,32,2Ø2,162,2Ø1         <18Ø>
123 DATA 49,1Ø4,42,136,2Ø8,245,24Ø,235,32,
    184,162,162,Ø,138,134,251,133,252       <Ø21>
124 DATA 168,32,2Ø7,255,2Ø1,58,176,132,233
    ,47,176,4,56,76,196,168,133,253         <Ø74>
125 DATA 6,251,38,252,165,252,133,254,165,
    251,1Ø,38,254,1Ø,38,254,24,1Ø1,251      <219>
126 DATA 8,24,1Ø1,253,17Ø,165,254,1Ø1,252,
    4Ø,1Ø5,Ø,76,52,169,32,122,162,234       <ØØØ>
127 DATA 234,234,234,162,4,189,135,16Ø,149
    ,17Ø,2Ø2,16,248,32,81,163,166,17Ø       <Ø7Ø>
128 DATA 165,171,32,2Ø5,221,23Ø,17Ø,2Ø8,2,
    23Ø,171,169,68,32,21Ø,255,169,193       <11Ø>
129 DATA 32,21Ø,255,16Ø,Ø,177,251,132,98,1
    33,99,32,2Ø9,221,32,99,164,162,3        <Ø75>
13Ø DATA 176,1Ø,169,44,166,211,224,73,144,
    227,162,9,134,198,189,125,16Ø,157       <166>
131 DATA 118,2,2Ø2,2Ø8,247,76,11Ø,163,32,1
    22,162,32,141,162,162,Ø,129,251         <222>
132 DATA 72,32,99,164,1Ø4,144,247,96,32,9Ø
    ,162,165,166,2Ø8,2,198,167,198,166      <186>
133 DATA 32,48,17Ø,134,181,16Ø,2,144,4,162
    ,2,16Ø,Ø,24,165,166,1Ø1,174,133         <ØØ8>
134 DATA 17Ø,165,167,1Ø1,175,133,171,161,1
    64,129,168,65,168,5,181,133,181         <Ø8Ø>
135 DATA 165,164,197,166,165,165,229,167,1
    76,29,24,181,164,121,1Ø7,16Ø,149        <162>
136 DATA 164,181,165,121,1Ø8,16Ø,149,165,1
    38,24,1Ø5,4,17Ø,2Ø1,7,144,232,233       <2Ø5>
137 DATA 8,17Ø,176,2Ø7,165,181,24Ø,15,76,2
    Ø9,162,56,162,254,181,17Ø,245,166       <111>
```

```
138 DATA 149,176,232,208,247,96,32,98,17Ø,
    76,214,169,76,98,17Ø,197,167,2Ø8        <195>
139 DATA 2,228,166,176,19,197,165,2Ø8,2,22
    8,164,144,11,133,18Ø,138,24,1Ø1         <Ø91>
14Ø DATA 174,17Ø,165,18Ø,1Ø1,175,96,32,9Ø,
    162,32,122,162,32,48,17Ø,32,2Ø3         <156>
141 DATA 164,2ØØ,169,16,36,171,24Ø,38,166,
    251,165,252,32,7Ø,17Ø,134,17Ø,177       <1Ø1>
142 DATA 251,133,181,32,74,165,16Ø,1,32,7Ø
    ,17Ø,2Ø2,138,24,229,17Ø,145,251         <ØØØ>
143 DATA 69,181,16,25,32,81,163,32,35,163,
    36,171,16,15,177,251,17Ø,2ØØ,177        <ØØ1>
144 DATA 251,32,7Ø,17Ø,145,251,138,136,145
    ,251,32,1Ø6,166,32,1Ø2,164,144,181      <Ø18>
145 DATA 96,32,1ØØ,162,162,39,32,64,163,32
    ,35,163,16Ø,6,162,Ø,32,76,163,161       <163>
146 DATA 251,32,57,164,2Ø8,249,162,Ø,32,93
    ,164,24Ø,3,76,186,17Ø,96,32,126         <148>
147 DATA 162,16Ø,Ø,32,2Ø7,255,234,234,234,
    32,2Ø2,162,2Ø1,46,24Ø,2,145,251         <245>
148 DATA 2ØØ,192,16,144,242,96,32,122,162,
    162,Ø,161,251,193,253,2Ø8,11,32         <2Ø1>
149 DATA 1Ø3,163,23Ø,253,2Ø8,243,23Ø,254,2
    Ø8,239,32,76,163,76,35,163,169,255      <172>
15Ø DATA 162,4,149,25Ø,2Ø2,2Ø8,251,32,2Ø2,
    162,162,5,221,11Ø,16Ø,24Ø,69,2Ø2        <Ø24>
151 DATA 2Ø8,248,134,169,32,18Ø,171,232,32
    ,2Ø7,255,2Ø1,32,24Ø,243,2Ø1,44,2Ø8      <11Ø>
152 DATA 3,32,122,162,32,81,163,164,169,17
    7,251,32,214,171,2Ø8,24,136,16,246      <223>
153 DATA 32,35,163,32,76,163,164,211,192,2
    Ø,144,9,32,148,164,32,114,164,32        <Ø44>
154 DATA 81,163,32,99,164,144,218,16Ø,19,7
    6,15Ø,164,189,115,16Ø,133,168,189       <199>
155 DATA 12Ø,16Ø,133,169,17Ø,24Ø,6,32,18Ø,
    171,2Ø2,2Ø8,25Ø,32,122,162,32,2Ø3       <1Ø2>
156 DATA 164,32,44,165,165,168,36,171,2Ø8,
    9,168,2Ø8,33,165,173,2Ø8,29,24Ø         <Ø92>
157 DATA 13,164,169,185,173,Ø,32,214,171,2
    Ø8,17,136,2Ø8,245,132,17Ø,32,14Ø        <212>
158 DATA 165,32,111,164,32,1Ø2,164,144,2Ø9
    ,96,32,1Ø6,166,24Ø,245,32,192,171       <134>
159 DATA 157,2Ø4,3,189,6Ø,3,157,1Ø8,3,32,2
    Ø2,162,16Ø,15,2Ø1,42,2Ø8,2,16Ø,Ø        <213>
16Ø DATA 32,175,162,157,6Ø,3,152,157,156,3
    ,96,133,18Ø,74,74,74,74,89,1Ø8,3        <245>
161 DATA 57,2Ø4,3,41,15,2Ø8,1Ø,165,18Ø,89,
    6Ø,3,57,156,3,41,15,96,1Ø4,1Ø4,32       <114>
162 DATA 2Ø7,255,2Ø1,87,2Ø8,3,76,86,173,2Ø
    1,66,2Ø8,3,76,2Ø8,173,2Ø1,81,2Ø8        <255>
163 DATA 3,76,79,173,2Ø1,83,24Ø,3,76,2Ø9,1
    62,32,141,162,72,32,141,162,72,32       <254>
164 DATA 73,162,16Ø,Ø,177,251,141,188,2,15
    2,145,251,169,54,141,22,3,169,172       <212>
165 DATA 141,23,3,162,252,76,236,163,162,3
    ,1Ø4,157,17Ø,2,2Ø2,16,249,1Ø4,1Ø4       <127>
166 DATA 186,142,174,2,173,168,2,133,252,1
    73,169,2,133,251,173,188,2,16Ø,Ø        <145>
167 DATA 145,251,169,2Ø,141,22,3,169,162,1
    41,23,3,169,82,76,255,162,32,81         <Ø66>
168 DATA 163,234,234,234,234,234,234,234,2
    34,96,141,171,2,8,1Ø4,41,239,141        <Ø95>
169 DATA 17Ø,2,142,172,2,14Ø,173,2,1Ø4,24,
    1Ø5,1,141,169,2,1Ø4,1Ø5,Ø,141,168       <144>
17Ø DATA 2,169,128,141,188,2,2Ø8,16,234,16
    2,12,32,246,252,216,162,5,1Ø4,157       <115>
171 DATA 168,2,2Ø2,16,249,173,2Ø,3,141,187
    ,2,173,21,3,141,186,2,186,142,174       <17Ø>
172 DATA 2,88,173,17Ø,2,41,16,24Ø,8,32,1Ø1
    ,172,169,82,76,255,162,44,188,2         <1Ø3>
173 DATA 8Ø,31,56,173,169,2,237,189,2,141,
    177,2,173,168,2,237,19Ø,2,13,177        <232>
174 DATA 2,2Ø8,1Ø3,173,191,2,2Ø8,95,169,12
    8,141,188,2,48,18,78,188,2,144,2Ø5      <1Ø7>
175 DATA 174,174,2,154,169,172,72,169,112,
    72,76,186,173,32,1Ø1,172,169,168        <178>
176 DATA 133,251,169,2,133,252,32,76,163,1
    6Ø,Ø,177,251,32,42,163,2ØØ,192,7        <Ø97>
177 DATA 24Ø,9,192,1,24Ø,242,32,76,163,2Ø8
    ,237,173,169,2,174,168,2,133,251        <144>
178 DATA 134,252,32,73,163,32,2Ø3,164,32,1
    99,165,32,228,255,24Ø,251,2Ø1,74        <183>
179 DATA 2Ø8,1Ø,169,1,141,188,2,2Ø8,47,2Ø6
    ,191,2,165,145,2Ø1,254,2Ø8,38,76        <Ø14>
18Ø DATA 189,172,32,242,173,169,64,2Ø8,1Ø,
    32,242,173,234,234,234,234,234,169      <Ø2Ø>
181 DATA 128,141,188,2,186,142,174,2,32,73
    ,162,32,1Ø1,172,173,188,2,24Ø,55        <Ø21>
182 DATA 162,Ø,234,234,234,234,234,234,234
    ,234,234,234,234,234,234,234,234        <131>
183 DATA 234,234,234,234,234,234,234,234,2
    34,12Ø,169,64,141,46,145,169,16Ø        <193>
184 DATA 141,46,145,169,71,141,4Ø,145,142,
    41,145,169,149,162,172,141,187,2        <ØØ7>
185 DATA 142,186,2,174,174,2,154,12Ø,173,1
    87,2,174,186,2,141,2Ø,3,142,21,3        <Ø42>
186 DATA 173,168,2,72,173,169,2,72,173,17Ø
    ,2,72,173,171,2,174,172,2,172,173       <234>
187 DATA 2,64,32,141,162,141,19Ø,2,32,141,
    162,141,189,2,32,141,162,141,191        <2Ø4>
188 DATA 2,76,214,162,173,184,2,174,185,2,
    141,2Ø,3,142,21,3,96,173,2Ø,3,174       <Ø14>
189 DATA 21,3,141,184,2,142,185,2,169,149,
    141,22,3,169,172,141,23,3,96,169        <Ø33>
19Ø DATA 31,141,15,144,234,234,234,234,162
    ,Ø,189,228,175,157,2Ø8,2,232,224        <Ø86>
191 DATA 13,144,245,162,42,32,64,163,32,2Ø
    7,255,2Ø1,42,24Ø,249,162,6,221,21Ø      <Ø73>
192 DATA 175,2Ø8,17,142,193,2,138,1Ø,17Ø,2
    32,189,216,175,72,2Ø2,189,216,175       <Ø42>
193 DATA 72,96,2Ø2,16,231,76,31,174,169,Ø,
    133,251,169,191,133,252,133,254         <142>
194 DATA 165,251,1Ø5,4,133,253,32,252,163,
    32,225,255,24Ø,15,173,141,2,24Ø         <156>
195 DATA 246,169,Ø,133,198,165,252,2Ø1,192
    ,144,227,76,31,174,32,126,162,16Ø       <Ø32>
196 DATA 18,162,Ø,32,2Ø2,162,32,154,162,12
    9,251,32,57,164,2Ø8,243,32,81,163       <Ø83>
197 DATA 76,36,174,32,85,175,173,193,2,2Ø1
    ,2,2Ø8,3,76,235,174,162,Ø,189,Ø         <227>
198 DATA 191,141,195,2,232,189,Ø,191,141,1
    96,2,138,76,2Ø3,174,32,194,162,2Ø8      <131>
199 DATA 3,76,141,174,32,141,162,141,195,2
    ,32,141,162,141,196,2,32,85,175         <Ø86>
2ØØ DATA 173,193,2,2Ø1,2,24Ø,32,32,13,175,
    162,13,32,198,255,16Ø,Ø,32,2Ø7,255      <243>
2Ø1 DATA 234,234,234,234,153,Ø,191,2ØØ,2Ø8
    ,243,32,2Ø4,255,32,188,175,76,73        <Ø43>
2Ø2 DATA 174,32,64,175,162,13,32,2Ø1,255,1
    6Ø,Ø,185,Ø,191,32,21Ø,255,166,144       <Ø58>
2Ø3 DATA 2Ø8,3,2ØØ,2Ø8,243,32,2Ø4,255,169,
    5Ø,32,13,175,76,182,175,141,2Ø9         <18Ø>
2Ø4 DATA 2,173,195,2,32,121,175,142,216,2,
    141,217,2,173,196,2,32,121,175,142      <111>
2Ø5 DATA 219,2,141,22Ø,2,162,15,32,2Ø1,255
    ,162,Ø,189,228,2,32,21Ø,255,232         <Ø34>
2Ø6 DATA 224,13,144,245,32,2Ø4,255,76,14Ø,
    175,162,15,32,2Ø1,255,162,Ø,189         <162>
2Ø7 DATA 242,175,32,21Ø,255,232,224,8,144,
    245,76,2Ø4,255,169,15,168,162,8         <Ø92>
2Ø8 DATA 32,186,255,169,Ø,32,189,255,32,19
    2,255,169,13,168,162,8,32,186,255       <251>
2Ø9 DATA 169,1,162,241,16Ø,175,32,189,255,
    76,192,255,162,48,56,233,1Ø,144         <146>
21Ø DATA 3,232,176,249,1Ø5,58,96,32,14Ø,17
    5,76,182,175,169,Ø,133,144,32,81        <144>
211 DATA 163,169,8,32,18Ø,255,169,111,32,1
    5Ø,255,32,165,255,2Ø1,48,2Ø8,6,76       <144>
212 DATA 171,255,32,165,255,32,21Ø,255,2Ø1
    ,13,2Ø8,246,32,171,255,1Ø4,1Ø4,32       <245>
213 DATA 188,175,76,31,174,169,13,32,195,2
    55,169,15,76,195,255,169,27,141         <132>
214 DATA 15,144,234,234,234,234,76,214,162
    ,58,82,87,77,88,64,114,174,172,174      <122>
215 DATA 172,174,72,174,197,175,133,175,85
    ,49,58,49,51,32,48,32,49,56,32,48       <217>
216 DATA 48,35,86,45,8Ø,32,49,51,32,48,183 <Ø31>
3ØØ OPEN 1,8,15:CLOSE 1:IF ST=Ø THEN 34Ø   <249>
31Ø PRINT"BITTE FLOPPY ANSCHALTEN, DANN <S
    PACE> "                                 <169>
32Ø GET T$:IF T$<>" "THEN 32Ø              <14Ø>
33Ø GOTO 3ØØ                                <Ø2Ø>
34Ø OPEN 8,8,8,"@Ø:SMON.EX VC2Ø,P,W"       <2ØØ>
35Ø PRINT#8,CHR$(Ø)CHR$(16Ø);              <144>
36Ø FOR I=Ø TO 4Ø9Ø:READ A:P=P+A:PRINT#8,C
    HR$(A);:NEXT                            <15Ø>
37Ø CLOSE 8                                 <183>
38Ø IF P<>523777 THEN PRINT"DATAS NICHT OK
    ":STOP                                  <ØØØ>
39Ø PRINT"DATAS OK"                         <Ø79>
```

Listing. »SMON VC 20« (Schluß)

Disketten-Monitor VC 20

Neben den hervorragenden Möglichkeiten, Disketteninhalte zu bearbeiten, läßt sich der Bildschirminhalt nach oben und unten scrollen. Außerdem stehen DOS-Erweiterungen, ähnlich dem DOS 5.1, zur Verfügung, die das Diskettenhandling auch vom Basic her wesentlich erleichtern.

Schreib-/Lesebefehle des Disketten-Monitors

Befehl	Beschreibung
R (Track Sektor) [Read Block]	liest den angegebenen Block in den Disketten-Puffer (liest den Block, auf den zuletzt zugegriffen wurde).
W (Track Sektor) [Write Block]	schreibt den Puffer in den angegebenen Block (schreibt den Puffer in den letzten Block).
N [Next Block]	liest den logisch nächsten Block in den Puffer. Nach dem letzten Block wird End of File! ausgegeben.
+	liest den physikalisch nächsten Block in den Puffer.
-	liest den physikalisch vorhergehenden Block in den Puffer.
S [Show]	zeigt aktuelle Track- und Sektornummern an.

Puffer bearbeiten

Befehl	Beschreibung
M (von bis) [Memory]	gibt den gesamten Pufferinhalt aus. (gibt den Pufferinhalt von . . . bis aus. Es können Adressen von $00 bis $FF angegeben werden.) Der Puffer kann durch Überschreiben geändert werden.
CRSR Up	der Puffer kann mit den Cursortasten gescrollt werden.
CRSR Down	Steht noch eine Adresse des Dumps auf dem Bildschirm, wird bei der nächsten (vorhergehenden) Adresse der Dump fortgesetzt. Ansonsten wird der Puffer von Anfang (Ende) ausgegeben.
P Adresse [Put Buffer to Memory]	der Puffer wird in den Speicher geschrieben. Dazu muß eine vierstellige Adresse angegeben werden. Es darf auch unter das Basic-ROM und unter das Kernal-ROM geschrieben werden, da dort der Puffer am wenigsten stört. Nicht zulässig ist der I/O-Bereich. Es werden jedoch keine Fehlermeldungen ausgegeben.
G Adresse [Get Buffer from Memory]	der Puffer wird aus dem angegebenen Bereich gelesen. Es gilt das gleiche wie beim P-Befehl.
C [Copy]	druckt den Pufferinhalt auf einem MPS 801 aus.

Die Diskbefehle

Befehl	Beschreibung
@ (Kommando)	liest den Fehlerkanal. (Sendet das angegebene Kommando zum Floppy-Laufwerk. Das Leerzeichen nach @ muß entfallen.)
$** oder **@$	listet das Directory. Es werden jeweils 20 Files gelistet. Dann stoppt der Ausdruck. Er kann nun mit RUN/STOP abgebrochen oder mit einer beliebigen Taste fortgesetzt werden.
B [BAM]	zeigt die BAM an. Alle Angaben werden in Hex ausgegeben, zum Beispiel Track 18 entspricht $12.
A Track Sektor [Allocate]	kennzeichnet den angegebenen Block in der BAM als belegt. Es wird anschließend zur Kontrolle der B-Befehl aufgerufen.
A T Track [Allocate Track]	kennzeichnet den angegebenen Track als belegt.
A A [Allocate All]	kennzeichnet die gesamte Diskette als belegt.
F Track Sektor [Free]	gibt den angegebenen Block frei.
F T Track [Free Track]	gibt den angegebenen Track frei.
F A [Free All]	gibt die gesamte Diskette frei.
T Track [Tracking]	zeigt die ersten 8 Byte eines jeden Sektors des angegebenen Tracks an.
X [Exit]	Disketten-Monitor verlassen

Unbenutzte Befehle

Befehl	Beschreibung
U	führt einen Kaltstart des Monitors aus (kann später für Erweiterungen benutzt werden).
H	es gilt das Gleiche wie unter U.

Das Dos

Nach dem Verlassen des Monitors werden die Befehle des DOS in der linken oberen Ecke des Bildschirms ausgegeben.

Befehl	Beschreibung
DLOAD	lädt ein Basic-Programm. Es kann direkt aus dem Directory ohne nachfolgenden Doppelpunkt geladen werden. Soll ein File absolut geladen werden, ist dies mit LOAD "Name",8,1 weiterhin möglich.
DSAVE	speichert ein Basic-Programm.
DVERIFY	vergleicht ein Programm mit dem Speicherinhalt.
DIR	listet das Directory ohne Programmverlust. Es gilt das bei »$« Gesagte. Anders als zum Beispiel in Simons Basic sind keine weiteren Eingaben zulässig.
DISK	sendet einen Befehl zum Floppy-Laufwerk. Der Befehl muß in Gänsefüßchen stehen.
DERROR	liest den Fehlerkanal.
DMON	zurück zum Disketten-Monitor.

Die Befehle des Disketten-Monitors

Das Vorprogramm des Disketten-Monitors (siehe Listing) erzeugt nach dem Start das Hauptprogramm auf Diskette. Dieses wird mit SYS 40960 gestartet und steht im Bereich von $A000 bis $AD6D.

Intern wird noch ein Puffer benötigt, auf den der Benutzer Zugriff hat. Der Puffer kann in den Speicher geschrieben und auch wieder gelesen werden, so daß die Möglichkeit besteht, ihn mit einem Maschinensprache-Monitor zu bearbeiten. Es werden keine Veränderungen am DOS der 1541 vorgenommen.

Soweit bekannt, ist der Disketten-Monitor mit allen Basic-Programmen und Hypra-Load kompatibel. Zu beachten ist lediglich, daß der Monitor den Interrupt-Vektor und das DOS den Basic-Vektor benutzt.

Die Befehle können entweder mit Parameter oder ohne Parameter eingegeben werden. Sind Parameter nicht erforderlich, werden sie im folgenden in Klammern angegeben (im Monitor keine Klammern eingeben). Ansonsten muß zwischen den Parametern ein Leerzeichen stehen. Die Angaben in eckigen Klammern sollen lediglich eine Eselsbrücke zum besseren Verständnis der Befehle darstellen. Alle Eingaben werden mit <RETURN> abgeschlossen. Nach Diskettenzugriffen werden automatisch eine oder mehrere (Fehler-)Meldungen des Floppy-Laufwerkes ausgegeben.

Der Disketten-Monitor ist natürlich nur von Disketten-Besitzern anzuwenden. Deshalb wird das Programm auch auf Diskette generiert; für Datasette ist ein ähnliches Programm nicht möglich. (K.-H. Templin/og)

```
0 DATA169,128,141,138,2,120,169,162,141,
20,3,169,160,141,21,3,88,169,0
1 DATA141,113,169,169,31,141,15,144,162,
216,160,169,32,58,169,32,95,168
2 DATA162,0,161,209,201,58,240,14,234,23
4,234,234,234,234,234,234,234,169
3 DATA46,32,210,255,169,0,141,109,169,32
,207,255,201,46,240,249,201,32
4 DATA240,245,162,19,221,115,169,208,18,
142,108,169,169,160,72,169,33,72
5 DATA189,135,169,72,189,155,169,72,96,2
02,16,230,76,34,160,169,58,76,210
6 DATA255,234,234,234,234,234,234,234,23
4,234,234,234,234,234,234,234,234
7 DATA234,234,234,234,234,234,234,234,23
4,234,234,234,234,234,234,234,234
8 DATA234,234,234,234,234,234,234,234,23
4,234,234,234,234,234,234,234,234
9 DATA234,169,160,72,169,230,72,8,72,72,
72,234,234,234,234,76,191,234,173
10 DATA113,169,208,17,120,169,162,141,20
,3,169,160,141,21,3,234,234,234
11 DATA234,234,88,96,173,113,169,208,250
,120,169,191,141,20,3,169,234,141
12 DATA21,3,234,234,234,234,234,234,234,
234,234,234,88,96,88,165,198,208
13 DATA3,76,24,235,173,119,2,201,17,208,
120,165,214,201,22,208,240,165,207
14 DATA240,6,169,0,133,198,240,230,165,2
09,133,92,165,210,133,93,169,23
15 DATA133,94,160,0,32,224,161,201,58,24
0,29,198,94,208,12,169,0,133,95
16 DATA133,198,32,10,162,76,76,161,56,16
5,92,233,22,133,92,176,222,198,93
17 DATA208,218,32,234,161,176,46,169,0,1
33,198,165,95,105,4,144,6,32,95
18 DATA168,76,95,161,168,32,117,168,152,
32,213,167,32,92,168,169,4,32,67
19 DATA168,32,234,168,162,22,160,1,32,12
,229,169,5,133,205,76,24,235,201
20 DATA145,208,249,165,214,208,245,165,2
07,240,6,169,0,133,198,240,235,165
21 DATA209,133,92,165,210,133,93,169,23,
133,94,160,0,32,224,161,201,58,240
22 DATA31,198,94,208,14,169,0,133,198,32
,32,162,32,10,162,169,252,208,32
23 DATA24,165,92,105,22,133,92,144,220,2
30,93,208,216,32,234,161,176,178
24 DATA169,0,133,198,32,32,162,56,165,95
,233,4,144,19,168,32,106,160,152
25 DATA32,213,167,32,92,168,169,4,32,67,
168,32,234,168,162,0,160,1,76,99
26 DATA161,177,92,200,201,32,176,2,9,64,
96,192,22,208,2,56,96,32,224,161
27 DATA201,32,240,243,32,33,168,10,10,10
,10,133,95,32,224,161,32,33,168
28 DATA5,95,133,95,169,255,133,204,165,2
07,240,10,165,206,164,211,145,209
29 DATA169,0,133,207,165,95,24,96,166,21
0,32,39,162,166,244,232,234,234
30 DATA134,173,134,97,162,0,134,172,169,
22,133,96,160,227,162,1,136,177
31 DATA172,145,96,152,208,248,198,173,19
8,97,202,16,241,169,32,166,210,134
32 DATA97,132,96,160,21,145,96,136,16,25
1,162,0,160,0,32,12,229,96,32,202
33 DATA160,169,128,141,113,169,169,27,14
1,15,144,234,234,234,234,234,160
34 DATA0,185,131,170,240,6,32,210,255,20
0,208,245,32,135,162,104,104,76
35 DATA103,228,169,146,141,8,3,169,162,1
41,9,3,96,32,115,0,201,68,240,6
36 DATA32,121,0,76,231,199,162,0,142,114
,169,160,1,189,96,170,240,8,209
37 DATA122,208,21,232,200,208,243,32,251
,200,173,114,169,10,170,189,118
38 DATA170,72,189,117,170,72,96,238,114,
169,173,114,169,201,7,240,11,189
39 DATA96,170,240,3,232,208,248,232,208,
203,160,0,240,188,169,1,44,169,0
40 DATA133,10,32,246,162,32,108,225,76,1
74,199,32,246,162,32,86,225,76,174
41 DATA199,169,0,32,189,255,162,8,160,0,
32,186,255,32,3,226,32,84,226,96
42 DATA32,253,225,138,168,162,8,76,186,2
55,32,117,164,76,174,199,76,246
43 DATA224,169,15,32,195,255,32,47,163,3
2,192,255,176,240,32,195,255,76
44 DATA174,199,169,0,32,189,255,169,15,1
68,162,8,32,186,255,32,3,226,32
45 DATA84,226,96,32,233,164,76,174,199,3
2,207,255,201,13,240,19,32,237,167
46 DATA144,60,141,80,170,32,57,168,32,23
7,167,144,49,141,83,170,32,108,163
47 DATA32,89,168,76,117,164,32,124,168,1
73,108,169,201,1,240,28,169,49,32
48 DATA176,168,162,13,32,198,255,162,0,3
2,207,255,157,0,175,232,208,247
49 DATA32,204,255,32,163,168,96,32,37,16
```

Listing. »Disketten-Monitor« für den VC 20

```
9,162,13,32,201,255,162,0,189,0
50 DATA175,32,210,255,232,208,247,32,204
,255,169,50,32,176,168,76,142,163
51 DATA174,0,175,240,12,173,1,175,142,80
,170,141,83,170,76,99,163,162,197
52 DATA160,170,76,58,169,174,80,170,173,
83,170,24,105,1,221,175,169,144
53 DATA226,169,0,232,224,36,208,219,162,
1,208,215,174,80,170,173,83,170
54 DATA56,233,1,176,204,202,208,2,162,35
,188,175,169,136,152,208,192,32
55 DATA207,255,201,13,240,90,32,237,167,
144,105,240,103,201,36,72,32,202
56 DATA160,169,128,141,113,169,104,141,8
0,170,169,0,141,83,170,162,213,160
57 DATA170,32,58,169,173,80,170,32,213,1
67,32,95,168,32,95,168,173,83,170
58 DATA32,213,167,32,89,168,32,75,169,16
9,4,160,0,32,67,168,32,234,168,32
59 DATA95,168,174,80,170,172,83,170,200,
152,221,175,169,176,6,141,83,170
60 DATA76,45,164,169,0,141,0,175,141,1,1
75,141,80,170,141,83,170,141,113
61 DATA169,32,179,160,96,32,207,255,201,
13,208,74,32,95,168,169,0,133,144
62 DATA32,202,160,169,8,133,186,32,180,2
55,169,111,133,185,32,150,255,32
63 DATA165,255,201,48,208,32,32,165,255,
201,48,208,8,169,145,32,210,255
64 DATA76,185,164,72,169,48,32,210,255,1
04,76,180,164,32,165,255,36,144
65 DATA112,5,32,210,255,208,244,32,171,2
55,76,179,160,201,36,208,3,76,233
66 DATA164,72,32,202,160,169,8,133,186,3
2,177,255,169,111,133,185,32,147
67 DATA255,104,32,168,255,32,207,255,201
,13,208,246,32,174,255,76,117,164
68 DATA32,95,168,32,202,160,169,11,133,2
53,169,0,133,144,169,62,133,187
69 DATA169,170,133,188,169,1,133,183,169
,8,133,186,169,96,133,185,32,149
70 DATA244,32,122,165,164,144,208,88,160
,6,132,251,32,165,255,166,252,133
71 DATA252,164,144,208,73,164,251,136,20
8,238,164,252,32,205,221,169,32
72 DATA32,210,255,32,165,255,166,144,208
,51,170,240,6,32,210,255,76,51,165
73 DATA32,95,168,198,253,208,31,32,171,2
55,169,11,133,253,32,179,160,165
74 DATA197,201,64,240,250,32,112,247,240
,14,32,122,165,164,144,208,7,32
75 DATA202,160,160,4,208,170,32,218,246,
32,179,160,169,0,133,198,76,120
76 DATA164,165,186,32,180,255,165,185,32
,150,255,96,32,237,167,144,19,168
77 DATA169,4,133,151,32,57,168,32,57,168
,32,25,169,208,248,32,234,168,96
78 DATA160,0,140,111,169,136,140,112,169
,32,207,255,201,13,240,21,32,237
79 DATA167,144,16,141,111,169,32,207,255
,201,13,240,6,32,237,167,141,112
80 DATA169,172,111,169,32,100,168,32,117
,168,152,32,213,167,32,92,168,169
81 DATA4,32,67,168,32,234,168,76,198,165
,32,207,255,201,13,240,34,201,32
82 DATA240,245,201,84,240,109,201,65,208
,3,76,159,166,32,2,168,144,96,141
83 DATA91,170,32,57,168,32,237,167,144,8
5,141,94,170,169,3,205,108,169,208
84 DATA3,169,65,44,169,70,141,87,170,32,
202,160,32,124,168,173,91,170,72
85 DATA32,44,168,142,91,170,141,92,170,1
73,94,170,72,32,44,168,142,94,170
86 DATA141,95,170,162,15,32,201,255,162,
0,189,85,170,32,210,255,232,224
87 DATA11,208,245,104,141,94,170,104,141
,91,170,32,204,255,32,163,168,32
88 DATA102,163,96,32,57,168,32,237,167,2
01,0,240,245,201,36,176,241,133
89 DATA95,32,193,166,169,3,205,108,169,2
08,3,24,144,1,56,8,165,95,72,32
90 DATA206,166,104,104,162,1,142,108,169
,32,99,163,162,1,134,198,162,13
91 DATA142,119,2,169,73,32,198,164,162,0
,142,113,169,76,246,170,32,193,166
92 DATA169,35,133,251,169,3,205,108,169,
208,3,24,144,1,56,8,165,251,72,32
93 DATA206,166,104,198,251,208,245,72,76
,127,166,32,202,160,162,18,169,0
94 DATA142,113,169,76,184,163,186,232,23
2,154,104,170,188,175,169,10,10
95 DATA170,40,176,4,169,0,240,25,152,157
,0,175,169,255,157,1,175,157,2,175
96 DATA152,56,233,17,168,185,211,169,157
,3,175,208,15,157,0,175,157,1,175
97 DATA157,2,175,157,3,175,240,1,104,186
,202,202,202,202,154,96,32,207,255
98 DATA201,13,240,34,32,58,167,32,202,16
0,234,234,234,234,234,177,251,153
99 DATA0,175,200,208,248,234,234,234,234
,32,179,160,169,0,141,80,170,141
100 DATA83,170,96,32,57,168,32,2,168,133
,252,32,57,168,32,2,168,133,251,160
101 DATA0,96,32,207,255,201,13,240,229,3
2,58,167,185,0,175,145,251,200,208
102 DATA248,76,49,167,169,4,170,160,0,32
,186,255,169,0,32,189,255,32,192
103 DATA255,162,4,162,4,32,201,255,76,20
2,160,32,95,168,32,95,168,162,226
104 DATA160,170,32,58,169,173,80,170,32,
213,167,162,235,160,170,32,58,169
105 DATA173,83,170,32,213,167,76,95,168,
32,207,255,201,13,240,9,201,36,240
106 DATA41,201,66,240,38,96,32,192,167,3
2,95,168,32,204,255,169,4,32,195
107 DATA255,76,179,160,160,255,140,112,1
69,200,140,111,169,32,98,167,32,124
108 DATA167,32,195,165,96,96,96,72,74,74
,74,74,32,23,168,170,104,41,15,32
109 DATA23,168,72,138,32,210,255,104,76,
210,255,169,0,141,110,169,32,57,168
110 DATA201,32,208,9,32,57,168,201,32,20
8,15,24,96,32,33,168,10,10,10,10
111 DATA141,110,169,32,57,168,32,33,168,
13,110,169,56,96,24,105,246,144,2
112 DATA105,6,105,58,96,201,58,8,41,15,4
0,144,2,105,8,96,162,48,56,233,10
113 DATA144,3,232,176,249,105,58,96,32,2
07,255,201,13,208,2,104,104,96,133
114 DATA151,32,92,168,185,0,175,32,213,1
67,200,208,3,238,109,169,198,151
115 DATA208,237,96,32,92,168,169,32,44,1
69,13,76,210,255,173,109,169,208
116 DATA6,204,112,169,176,1,96,104,104,9
6,32,95,168,169,58,162,13,76,228
117 DATA167,32,202,160,169,15,168,162,8,
```

```
32,186,255,169,0,32,189,255,32,192
118 DATA255,169,13,168,162,8,32,186,255,
169,1,162,63,160,170,32,189,255,76
119 DATA192,255,169,13,32,195,255,169,15
,32,195,255,76,179,160,141,73,170
120 DATA173,80,170,72,32,44,168,142,80,1
70,141,81,170,173,83,170,72,32,44
121 DATA168,142,83,170,141,84,170,162,15
,32,201,255,162,0,189,72,170,32,210
122 DATA255,232,224,13,208,245,104,141,8
3,170,104,141,80,170,76,204,255,152
123 DATA56,233,4,168,32,92,168,169,18,32
,210,255,162,4,185,0,175,41,127,201
124 DATA32,176,4,169,46,208,3,185,0,175,
32,210,255,169,0,133,212,200,202
125 DATA208,229,169,146,76,210,255,32,23
7,167,144,3,153,0,175,200,198,151
126 DATA96,162,15,32,201,255,162,0,189,6
4,170,32,210,255,232,224,8,208,245
127 DATA76,204,255,134,251,132,252,160,0
,177,251,240,6,32,210,255,200,208
128 DATA246,96,32,124,168,169,49,32,176,
168,162,13,32,198,255,162,0,32,207
129 DATA255,157,0,175,232,224,4,208,245,
32,204,255,76,163,168,255,13,0,0
130 DATA144,164,128,6,58,87,82,65,70,78,
64,36,77,71,80,83,67,88,66,85,72
131 DATA43,45,84,165,163,163,165,165,163
,164,164,165,167,167,167,167,162
132 DATA170,159,159,163,163,163,132,72,7
2,221,221,175,109,232,157,15,76,123
133 DATA158,95,245,255,255,199,224,247,0
,21,21,21,21,21,21,21,21,21,21,21
134 DATA21,21,21,21,21,21,19,19,19,19,19
,19,19,18,18,18,18,18,18,17,17,17
135 DATA17,17,1,3,7,31,31,147,31,18,49,5
3,52,49,32,68,73,83,75,45,77,79,78
136 DATA73,84,79,82,32,86,50,46,53,13,40
,67,41,32,66,89,32,72,79,82,83,84
137 DATA32,82,69,73,67,72,69,82,84,13,32
,32,32,32,32,53,57,48,48,32,83,73
138 DATA69,71,69,78,13,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
139 DATA0,0,0,0,0,0,0,0,0,0,0,0,36,35,66
,45,80,32,49,51,32,48,85,49,58,49
140 DATA51,32,48,32,18,56,32,0,48,66,45,
70,32,48,32,1,49,32,0,48,147,0,148
141 DATA0,149,0,73,83,75,0,73,82,0,69,82
,82,176,0,77,145,0,223,162,236,162
142 DATA220,162,27,163,56,163,18,163,255
,159,147,154,18,68,76,79,65,68,32
143 DATA32,13,18,68,83,65,86,69,32,32,13
,18,68,86,69,82,73,70,89,13,18,68
144 DATA73,82,32,32,32,32,13,18,68,73,83
,75,32,32,32,13,18,68,69,82,82,79
145 DATA82,32,13,18,68,77,79,78,32,32,32
,13,0,32,32,69,78,68,32,79,70,32
146 DATA70,73,76,69,32,33,0,147,32,84,82
,65,67,75,32,58,32,0,32,32,84,82
147 DATA65,67,75,32,58,32,0,32,83,69,75,
84,79,82,32,58,32,0,32,4,171,162
148 DATA0,142,113,169,32,179,160,76,30,1
72,32,193,166,162,41,160,172,32,58
149 DATA169,162,144,189,0,175,32,210,255
,232,224,160,208,245,32,89,168,162
150 DATA95,160,172,32,58,169,173,162,175
,32,210,255,173,163,175,32,210,255
151 DATA32,30,172,32,117,172,234,234,234
,160,20,152,32,213,167,169,45,32
152 DATA210,255,152,240,3,32,95,168,136,
152,16,238,162,0,160,3,32,12,229
153 DATA160,2,169,48,162,15,208,10,234,2
34,234,234,234,234,234,234,162,3
154 DATA32,210,255,202,208,250,24,105,1,
136,208,242,162,1,160,3,32,12,229
155 DATA160,18,169,1,133,251,32,23,168,3
2,210,255,230,251,165,251,201,16
156 DATA208,4,169,0,133,251,136,208,235,
169,1,133,2,162,2,165,2,105,2,168
157 DATA32,12,229,162,20,134,139,169,175
,133,141,169,1,166,2,105,4,202,208
158 DATA251,133,140,165,139,166,2,224,18
,144,32,224,25,144,20,224,31,144
159 DATA8,201,17,144,20,162,4,208,42,201
,18,144,12,162,4,208,34,201,19,144
160 DATA4,162,4,208,26,165,139,74,74,74,
168,177,140,168,165,139,41,7,170
161 DATA152,61,102,172,208,4,162,0,240,2
,162,2,160,2,189,110,172,32,210,255
162 DATA232,136,208,246,165,139,240,5,16
9,17,32,210,255,198,139,16,165,230
163 DATA2,165,2,201,19,240,103,76,154,17
1,96,169,0,133,198,165,198,240,252
164 DATA198,198,96,147,42,61,66,69,76,69
,71,84,13,46,61,70,82,69,73,13,166
165 DATA61,76,69,69,82,13,13,68,73,83,75
,78,65,77,69,32,58,32,0,0,0,0,0,0
166 DATA0,0,0,0,0,0,0,0,0,0,0,13,73,68
,32,58,32,0,1,2,4,8,16,32,64,128
167 DATA42,157,46,157,166,157,96,169,147
,32,210,255,32,95,168,32,95,168,96
168 DATA32,30,172,32,117,172,160,20,152,
32,213,167,169,45,32,210,255,152
169 DATA240,3,32,95,168,136,152,16,238,1
62,0,160,3,32,12,229,160,2,169,49
170 DATA162,13,208,10,234,234,234,234,23
4,234,234,234,162,4,32,210,255,202
171 DATA208,250,24,105,1,136,208,242,162
,1,160,3,32,12,229,160,17,169,3,133
172 DATA251,32,23,168,32,210,255,230,251
,165,251,201,16,208,4,169,0,133,251
173 DATA136,208,235,169,1,133,3,162,2,16
5,3,105,2,168,32,12,229,162,20,134
174 DATA139,169,175,133,141,169,1,166,2,
105,4,202,208,251,133,140,165,139
175 DATA166,2,224,18,144,32,224,25,144,2
0,224,31,144,8,201,17,144,20,162
176 DATA4,208,42,201,18,144,12,162,4,208
,34,201,19,144,4,162,4,208,26,165
177 DATA139,74,74,74,168,177,140,168,165
,139,41,7,170,152,61,102,172,208
178 DATA4,162,0,240,2,162,2,160,2,189,11
0,172,32,210,255,232,136,208,246
179 DATA165,139,240,5,169,17,32,210,255,
198,139,16,165,230,2,230,3,165,3
180 DATA201,18,240,3,76,231,172,96
200 OPEN8,8,8,"DISK-MONITOR VC20,P,W"
210 PRINT#8,CHR$(0)CHR$(160);
220 FORI=0TO3436:READA:P=P+A:PRINT#8,CHR
$(A);:NEXT
230 CLOSE8
240 IFP<>420282THENPRINT"DATAS NICHT OK!
":STOP
250 PRINT"DATAS OK"
```

Listing. »Disketten-Monitor« für den VC 20 (Schluß)

40 Zeichen auf dem VC 20

Gut lesbare 40 Zeichen pro Zeile sind ein alter Wunsch für VC 20-Besitzer. Durch einen frei definierbaren Zeichensatz ist das Programm selbst 40-Zeichen-Karten überlegen. 10 neue Befehle unterstützen die Programmierung.

Extended Screen V 4.0 wird vom DATA-Lader (siehe Listing) im Modulbereich ab $A000 (RAM-Erweiterung!) abgelegt und setzt daher einen voll ausgebauten VC 20 voraus. Das Programm ergänzt das Commodore Basic um insgesamt 10 neue Befehle, die alle dazu dienen, die Bildschirmdarstellung des VC 20 auf 22 Zeilen à 40 Zeichen zu erweitern (Bild 1 und 2). Hierzu werden im Bereich 1000 bis 1F00 ein Grafikbildschirm und eine Bit-Map eingerichtet, in die die Zeichen später hineinkopiert werden. Die neuen Zeichen erhalten dabei eine Matrix von 8 x 4 Bildpunkten, sind also genauso hoch, jedoch nur halb so breit wie die Zeichen des normalen Bildschirms.

Um die neuen Befehle ins Basic einzubinden, wird der Vektor für Basic-Befehlsadresse (0308/0309) auf eine Routine gerichtet, die vor der Ermittlung einer Befehlsadresse zunächst überprüft, ob ein Befehl mit vorangestelltem Ausrufungszeichen vorliegt. Ist dies der Fall, so wird in eine neue Auswertungsroutine verzweigt. Ebenfalls verändert werden müssen: der Vektor für Fehlermeldung (0300/0301), der IRQ-Vektor (0314/0315), der BRK-Vektor (0316/0317) und der NMI-Vektor (0318/0319). Die neue Routine zur Fehlerausgabe löscht vor jeder Meldung zunächst den Bildschirmspeicher und schaltet in den 22-Zeichen-Modus um, damit eventuelle Fehler nicht dadurch unerkannt bleiben, daß sich die entsprechende Meldung irgendwo in der Bit-Map des 40-Zeichen-Modus verliert. Die neue IRQ-Routine hält den Basic-Start ständig über 1F00, um Basic-Programme so vor einer Zerstörung durch das Einschalten des 40-Zeichen-Modus zu schützen.

Die neuen BRK- und NMI-Routinen dienen nur dazu, ein Verändern des IRQ-Vektors beim <RUN/STOP + RESTORE> zu verhindern. Der letzte Vektor, den das Programm verändert, ist der Ausgabe-Vektor (0326/0327). Sobald der 40-Zeichen-Modus eingeschaltet wird, wird dieser Vektor auf eine Routine gerichtet, die einen »?BAD MODE ERROR« ausgibt, wenn irgendein Standart-Basic-Befehl aufgerufen wird, der normalerweise eine Bildschirmausgabe zur Folge hätte (PRINT, INPUT, etc.).

Die neuen Befehle

!F (Fourty)
Schaltet in den 40-Zeichen-Modus um. In diesem Modus darf nur mit Hilfe der neuen Befehle gedruckt werden! Dazu steht jedoch nicht der gesamte Zeichensatz des VC 20 zur Verfügung. Der 40-Zeichen-Modus kennt lediglich die CHR$-Codes 32-95, 18 und 146. Dieses Manko wird jedoch durch den Befehl !D mehr als ausgeglichen.

!T (Twenty)
Schaltet zurück in den 22-Zeichen-Modus. Bei diesem, wie auch bei dem !F-Befehl sollte man gleichzeitig die Befehle !H und !S einsetzen, um ein flimmerfreies Umschalten zu erzielen.

!H (Hide)
Schaltet den Bildschirm ab.

!S (Show)
Schaltet den Bildschirm wieder ein und zentriert ihn je nachdem, ob er im 40- oder 22-Zeichen-Modus eingesetzt wird.

!P (Print) . . . !Px,y,string
Schreibt den »string« im 40-Zeichen-Modus an die Bildschirmposition »x/y«. Für »x« darf entweder eine Zahl zwischen 0 und 39, eine Variable oder ein Ausdruck stehen. Für »y« gilt das gleiche, nur muß hier die Zahl zwischen 0 und 21 liegen. Für »string« steht entweder ein Text in Anführungszeichen, eine Stringvariable oder ein Ausdruck. Zahlen müssen über die STR-Funktion in einen String umgewandelt werden. Wird !P im 22-Zeichen-Modus eingesetzt, so quittiert der Computer dies mit einem »?BAD MODE ERROR«.

!M (Middle) . . . !My,string
Druckt den »string« im 40-Zeichen-Modus in der in »y« angegebenen Zeile zentrisch, also genau in der Zeilenmitte. Für »y« und »string« gilt das gleiche wie unter !P erwähnt. Auch dieser Befehl verursacht im 22-Zeichen-Modus einen »?BAD MODE ERROR«. Der String darf höchstens 40 Zeichen lang sein!

!I (Inkey) . . . !Ix, y,e,stringvariable
Mit diesem Befehl können Texteingaben im 40-Zeichen-Modus vorgenommen werden. Für x und y gilt das gleiche wie unter !P erwähnt. Für »l« steht eine Zahl, eine numerische Variable oder ein Ausdruck. »l« gibt die maximale erlaubte Länge des einzugebenden Strings an. »stringvariable« steht für die Variable, in der der eingegebene String später stehen soll. Für !I gilt genau wie für alle anderen Befehle, daß ein auszudruckender String nie den rechten Bildschirmrand überschreiten darf.

Beim !I-Befehl werden alle Tasten mit CHR$-Code 32-95, 20 und 13 akzeptiert; eine Cursormanipulation ist also nicht möglich. Ist bei einer Eingabe die in »l« angegebene Höchstlänge erreicht, so werden nur noch die Tasten <RETURN> und <DEL> akzeptiert! Zu beachten ist noch, daß die Summe aus x + l immer kleiner als 40 sein muß!

!C (Color) . . . !Cf
Verändert die Druckfarbe entsprechend »f«, wobei für »f« eine Zahl zwischen 0 und 7, eine numerische Variable oder ein Ausdruck stehen kann. Zu beachten ist, daß die Druckfarbe immer auf dem gesamten Bildschirm gleich ist. Beim Umschalten in den 40-Zeichen-Modus wird eine Druckfarbe dem Wert in 0286 (aktueller Farbcode) entsprechend eingestellt.

!D (Define) . . . !Dz,w1,w2,w3,w4,w5,w6,w7,w8
Hierbei handelt es sich um einen Befehl, den wohl die wenigsten 40-Zeichen-Karten bieten dürften. Er ermöglicht das Definieren eigener Sonderzeichen für den 40-Zeichen-Modus! Für »z« steht entweder eine Zahl zwischen 32 und 95, eine numerische Variable oder ein Ausdruck. »z« gibt den CHR$-Code des Zeichens, das umdefiniert werden soll. »w1« bis »w8« sind jeweils Zahlen zwischen 0 und 15, numerische Variablen oder Ausdrücke. Sie geben die neue Matrix

des Zeichens an und errechnen sich genau wie beim Umdefinieren eines Zeichens des normalen Zeichensatzes.

!R (Re-define)

Stellt den normalen Zeichensatz der 40-Zeichen-Karte wieder her.

Frei definierbarer Zeichensatz

Für die Assembler-Freaks sei noch erwähnt, wie man die neuen Befehle auch mit Assembler nutzen kann. Dies ist leider nicht bei jedem Befehl direkt möglich. Die Inkey-Routine zum Beispiel endet mit dem Übertragen des eingegebenen Textes in die angegebene Variable, deren Name zu diesem Zweck aus dem Basic-Text gelesen wird. Wenn man diese Routine nun vom Assembler aufrufen würde, so würde sie immer mit einem »?SYNTAX ERROR« enden, da ja kein Basic-Text vorhanden wäre, aus dem sie einen Variablennamen lesen könnte. Ähnlich verhält es sich auch mit dem Define-Befehl. Er liest während der Abarbeitung immer wieder die zur Umdefinierung eines Zeichens nötigen Werte aus dem Basic-Text.

Doch nun zu nutzbaren Routinen:

Umschalten in den 40-Zeichen-Modus: JSR $ A10E

Umschalten in den 22-Zeichen-Modus: JSR $ A18C

Bildschirm abschalten: JSR $ A1B6

Bildschirm einschalten: JSR $ A1BF

Druckfarbe ändern:

LDA #$ (Farbcode 00 bis 07)

JSR $ A174

String an Position x/y ausgeben:

LDX #$ (X-Koordinate)

LDY #$ (Y-Koordinate)

STX $ 02AA

STY $ 02AB

JSR $ A415

LDX $ 62

LDY $ 63

STX $ 02AE

STY $ 02AF

LDA #$ (Länge des auszugebenden Strings)

LDX #$ (LSB der Adresse, an der der String im Speicher steht)

LDY #$ (MSB der Adresse, an der der String im Speicher steht)

JSR $ A4B4

String zentrisch in Zeile y ausgeben:

LDY #$ (Y-Koordinate)

STY $ 02AB

LDA #$ (Länge des auszugebenden Strings)

LDX #$ (LSB der Adresse, an der der String im Speicher steht)

LDY #$ (MSB der Adresse, an der der String im Speicher steht)

JSR $ A5E3

Normalen Zeichensatz wieder herstellen:

JSR $ A675

(M. Fichtner/og)

```
10 PRINT"{CLR,2DOWN,RIGHT}EXTENDED SCREEN
   V4.0":PRINT"{3DOWN,4RIGHT,2SPACE}APRIL
   1985"                                    <161>
20 PRINT"{3DOWN,RIGHT}BY MATTHIAS FICHTNER
   ":PRINT"{DOWN,4RIGHT}LORTZINGSTR.10C"    <053>
30 PRINT"{DOWN,4RIGHT}6729 WOERTH/RH.1":PR
   INT"{DOWN,4RIGHT}TEL.:07271/6622"        <090>
40 PRINT"{3DOWN,RIGHT}PLEASE WAIT A MOMENT
   "                                        <010>
50 FOR T=0 TO 1972:READ B:Z=Z+B:NEXT:IF Z<
   >225685 THEN PRINT"{CLR,DOWN,4RIGHT}ERR
   OR IN DATAS":END                         <008>
60 RESTORE:FOR T=0 TO 1972:READ B:POKE 409
   60+T,B:NEXT:PRINT"{UP,5SPACE}HIT 'CTRL'
   !{5SPACE}"                               <195>
70 WAIT 653,4:PRINT"{CLR}";:SYS 40960       <066>
100 DATA 120,169,80,141,0,3,169,160,141,1,
    3,169,137,141,8,3,169,160,141,9         <238>
101 DATA 3,169,67,141,20,3,169,160,141,21,
    3,169,213,141,22,3,169,254,141,23       <061>
102 DATA 3,169,93,141,24,3,169,160,141,25,
    3,169,31,133,44,169,0,141,0,31,32       <013>
103 DATA 68,198,88,76,116,196,165,44,201,3
    1,176,4,169,31,133,44,76,191,234        <192>
104 DATA 138,72,32,128,161,32,191,161,104,
    170,76,58,196,72,138,72,152,72,173      <113>
105 DATA 29,145,16,28,45,30,145,170,41,2,2
    40,23,44,17,145,32,52,247,32,225        <130>
106 DATA 255,208,9,32,249,253,32,24,229,10
    8,2,192,76,255,254,76,222,254,32        <126>
107 DATA 115,0,201,33,240,6,32,121,0,76,23
    1,199,32,115,0,170,32,115,0,224         <218>
108 DATA 70,208,6,32,14,161,76,174,199,224
    ,72,208,6,32,182,161,76,174,199         <007>
109 DATA 224,83,208,6,32,191,161,76,174,19
    9,224,84,208,6,32,128,161,76,174        <232>
110 DATA 199,224,67,208,6,32,182,165,76,17
    4,199,224,77,208,6,32,202,165,76        <241>
111 DATA 174,199,224,80,208,6,32,147,164,7
    6,174,199,224,68,208,6,32,21,166        <059>
112 DATA 76,174,199,224,82,208,6,32,117,16
    6,76,174,199,224,73,208,6,32,180        <057>
113 DATA 166,76,174,199,76,4,207,234,234,2
    34,234,234,234,234,234,234,234,32       <195>
114 DATA 169,165,169,128,141,145,2,169,147
    ,32,210,255,169,14,141,0,144,169        <248>
115 DATA 20,141,2,144,169,151,141,3,144,16
    9,40,141,166,2,169,224,141,60,161       <229>
116 DATA 169,16,141,61,161,169,0,141,0,31,
    238,60,161,208,248,238,61,161,172       <148>
117 DATA 61,161,192,31,144,238,162,0,169,1
    4,157,0,16,232,168,200,152,224,220      <202>
118 DATA 208,245,169,204,141,5,144,169,220
    ,141,38,3,169,161,141,39,3,173,134      <139>
119 DATA 2,201,8,144,2,169,6,162,221,157,2
    55,147,202,208,250,96,234,234,234       <098>
120 DATA 169,0,141,145,2,169,122,141,38,3,
    169,242,141,39,3,169,147,32,210         <240>
121 DATA 255,169,22,141,2,144,169,174,141,
    3,144,169,192,141,5,144,169,12,141      <028>
122 DATA 0,144,169,22,141,2,144,169,0,141,
    166,2,96,234,234,234,169,255,141        <047>
123 DATA 1,144,96,234,234,234,173,166,2,24
    0,4,169,40,208,2,169,38,141,1,144       <063>
124 DATA 96,234,234,234,13,63,66,65,68,32,
    77,79,68,69,0,32,128,161,32,191         <118>
125 DATA 161,169,209,160,161,32,30,203,76,
    101,196,234,234,234,0,0,0,0,0,0         <092>
126 DATA 0,0,68,68,68,68,68,0,68,0,170,170
    ,0,0,0,0,0,0,0,170,238,170,238,170      <083>
127 DATA 0,0,68,238,136,238,34,238,68,0,17
    0,34,68,68,68,136,170,0,238,136         <221>
128 DATA 68,68,68,136,238,0,68,136,0,0,0,0
    ,0,0,34,68,68,68,68,68,34,0,136         <253>
129 DATA 68,68,68,68,68,136,0,0,170,238,68
    ,238,170,0,0,0,0,68,238,68,0,0,0        <126>
130 DATA 0,0,0,0,0,68,68,136,0,0,0,238,0,0
    ,0,0,0,0,0,0,0,0,68,0,34,34,68,68       <144>
131 DATA 68,136,136,0,68,170,170,170,238,1
    70,68,0,68,204,68,68,68,68,238,0        <043>
132 DATA 68,170,34,68,136,136,238,0,68,170
    ,34,68,34,170,68,0,136,136,170,238      <175>
133 DATA 34,34,34,0,238,136,136,204,34,34,
    204,0,68,170,136,204,170,170,68         <239>
```

Listing. »Extended Screen V 4.0« für den VC 20

```
134 DATA 0,238,34,34,68,68,68,68,0,68,170,
    170,68,170,170,68,0,68,170,170,102        <011>
135 DATA 34,170,68,0,0,0,68,0,68,0,0,0,0,0
    ,0,68,0,68,68,136,0,34,68,136,68          <181>
136 DATA 34,0,0,0,0,238,0,238,0,0,0,0,136,
    68,34,68,136,0,0,68,170,34,68,68          <035>
137 DATA 0,68,0,68,170,170,204,136,170,68,
    0,68,170,170,238,170,170,170,0,204        <093>
138 DATA 170,170,204,170,170,204,0,68,170,
    136,136,136,170,68,0,204,170,170          <225>
139 DATA 170,170,170,204,0,238,136,136,204
    ,136,136,238,0,238,136,136,204,136        <098>
140 DATA 136,136,0,68,170,136,170,170,170,
    68,0,170,170,170,238,170,170,170          <117>
141 DATA 0,238,68,68,68,68,68,238,0,238,34
    ,34,34,34,170,68,0,170,170,170,204        <132>
142 DATA 170,170,170,0,136,136,136,136,136
    ,136,238,0,170,238,238,170,170,170        <058>
143 DATA 170,0,238,170,170,170,170,170,170
    ,0,68,170,170,170,170,170,68,0,204        <113>
144 DATA 170,170,204,136,136,136,0,68,170,
    170,170,170,238,102,0,204,170,170         <009>
145 DATA 204,170,170,170,0,68,170,136,68,3
    4,170,68,0,238,68,68,68,68,68,68          <138>
146 DATA 0,170,170,170,170,170,170,68,0,17
    0,170,170,170,170,68,68,0,170,170         <099>
147 DATA 170,170,238,238,170,0,170,170,170
    ,68,170,170,170,0,170,170,170,238         <045>
148 DATA 68,68,68,0,238,34,34,68,136,136,2
    38,0,102,68,68,68,68,68,102,0,102         <081>
149 DATA 68,68,238,68,204,238,0,204,68,68,
    68,68,68,204,0,0,0,68,170,170,170         <111>
150 DATA 0,0,0,0,102,136,102,0,0,0,234,234
    ,234,32,169,165,32,121,0,32,158           <139>
151 DATA 215,224,40,176,16,142,170,2,32,25
    3,206,32,121,0,32,158,215,224,22          <009>
152 DATA 144,3,76,72,210,142,171,2,169,224
    ,133,98,169,16,133,99,138,74,170          <109>
153 DATA 232,202,240,16,165,98,24,105,64,1
    33,98,144,2,230,99,230,99,76,33           <143>
154 DATA 164,173,171,2,201,2,144,6,56,233,
    2,76,55,164,10,10,10,141,171,2,165        <180>
155 DATA 98,24,109,171,2,133,98,144,2,230,
    99,173,170,2,74,133,100,169,0,133         <162>
156 DATA 101,6,100,38,101,6,100,38,101,6,1
    00,38,101,6,100,38,101,165,98,24          <209>
157 DATA 101,100,133,98,144,2,230,99,165,9
    9,24,101,101,133,99,173,170,2,201         <197>
158 DATA 2,144,6,56,233,2,76,130,164,141,1
    71,2,96,234,234,234,32,169,165,173        <101>
159 DATA 166,2,208,3,76,220,161,32,242,163
    ,165,98,141,174,2,165,99,141,175          <237>
160 DATA 2,32,253,206,32,154,205,32,163,21
    4,141,167,2,134,100,132,101,173           <168>
161 DATA 174,2,133,98,173,175,2,133,99,174
    ,167,2,138,240,196,24,109,170,2           <178>
162 DATA 201,41,144,5,162,23,76,55,196,160
    ,0,132,107,177,100,201,32,144,4           <247>
163 DATA 201,96,144,3,76,95,165,56,233,32,
    133,106,6,106,38,107,6,106,38,107         <229>
164 DATA 6,106,38,107,165,106,24,105,239,1
    33,106,144,2,230,107,165,107,24           <181>
165 DATA 105,161,133,107,160,7,177,106,41,
    15,166,199,240,2,73,15,133,108,173        <156>
166 DATA 171,2,208,19,165,108,10,10,10,10,
    133,108,177,98,41,15,5,108,145,98         <098>
167 DATA 76,58,165,177,98,41,240,5,108,145
    ,98,136,16,209,206,167,2,230,100          <087>
168 DATA 208,2,230,101,169,1,56,237,171,2,
    141,171,2,208,11,165,98,24,105,16         <114>
169 DATA 133,98,144,2,230,99,76,197,164,20
    1,18,240,11,201,146,240,23,201,147        <193>
170 DATA 240,26,76,61,165,169,1,133,199,20
    6,167,2,230,100,208,2,230,101,76          <200>
171 DATA 197,164,169,0,133,199,76,114,165,
    169,224,141,146,165,169,16,141,147        <011>
172 DATA 165,169,0,141,0,31,238,146,165,20
    8,248,238,147,165,172,147,165,192         <092>
173 DATA 31,144,238,76,114,165,234,234,234
    ,165,157,240,5,162,21,76,55,196           <104>
174 DATA 96,234,234,234,32,163,166,32,158,
    215,138,201,8,144,3,76,72,210,76          <187>
175 DATA 116,161,234,234,234,32,163,166,32
    ,158,215,224,22,144,3,76,72,210           <225>
176 DATA 142,171,2,32,253,206,32,154,205,3
    2,163,214,141,167,2,142,168,2,140         <190>
177 DATA 169,2,201,41,144,5,162,23,76,55,1
    96,169,40,56,237,167,2,74,141,170         <202>
178 DATA 2,174,171,2,32,21,164,173,168,2,1
    33,100,173,169,2,133,101,76,197           <222>
179 DATA 164,234,234,234,32,121,0,32,158,2
    15,224,32,176,3,76,72,210,224,96          <191>
180 DATA 176,249,169,0,133,88,138,56,233,3
    2,133,87,6,87,38,88,6,87,38,88,6          <240>
181 DATA 87,38,88,165,87,24,105,239,133,87
    ,144,2,230,88,165,88,24,105,161           <132>
182 DATA 133,88,160,0,177,87,41,240,145,87
    ,152,72,32,253,206,32,158,215,104         <091>
183 DATA 168,224,16,144,3,76,72,210,138,17
    ,87,145,87,200,192,8,144,223,96           <092>
184 DATA 234,234,234,169,239,133,97,169,16
    1,133,98,160,0,177,97,41,240,145          <228>
185 DATA 97,74,74,74,74,17,97,145,97,230,9
    7,208,2,230,98,165,98,201,163,144         <194>
186 DATA 230,165,97,201,239,144,224,96,234
    ,234,234,32,169,165,173,166,2,208         <214>
187 DATA 3,76,220,161,76,121,0,234,234,234
    ,32,163,166,32,248,163,165,98,141         <049>
188 DATA 174,2,165,99,141,175,2,173,171,2,
    141,176,2,32,253,206,32,121,0,32          <037>
189 DATA 158,215,224,40,144,3,76,72,210,13
    8,24,109,170,2,201,40,176,244,142         <040>
190 DATA 177,2,169,0,162,39,157,0,2,202,16
    ,250,141,178,2,173,176,2,141,171          <022>
191 DATA 2,162,0,160,2,173,178,2,32,180,16
    4,32,108,167,173,178,2,205,177,2          <155>
192 DATA 176,29,32,228,255,240,251,201,96,
    176,247,201,32,176,3,76,156,167           <079>
193 DATA 174,178,2,157,0,2,232,142,178,2,7
    6,243,166,32,228,255,240,251,201          <041>
194 DATA 20,240,115,201,13,208,243,32,134,
    167,32,253,206,32,139,208,166,13          <152>
195 DATA 240,39,133,100,132,101,173,178,2,
    32,125,212,172,178,2,185,0,2,145          <025>
196 DATA 98,136,16,248,160,0,173,178,2,145
    ,100,200,165,98,145,100,200,165           <190>
197 DATA 99,145,100,96,160,7,177,98,174,17
    1,2,208,7,41,15,9,240,76,128,167          <020>
198 DATA 41,240,9,15,145,98,136,16,233,96,
    160,7,177,98,174,171,2,208,5,41           <192>
199 DATA 15,76,150,167,41,240,145,98,136,1
    6,237,96,201,20,240,7,201,13,240          <007>
200 DATA 148,76,6,167,173,178,2,240,6,32,1
    34,167,206,178,2,76,243,166               <182>
```

Listing. »Extended Screen V 4.0« für 40-Zeichen-Darstellung auf dem VC 20 (Schluß).
Beachten Sie die Eingabehinweise auf Seite 129.

Inserentenverzeichnis

Tips & Tricks zum C 16/C 116

In dieser kleinen Sammlung von Tips und Tricks stellen wir Ihnen unter anderem nützliche PEEKs und POKEs, ein Programm zur Veränderung des Zeichensatzes, eine kurze Hardcopy-Routine zum Ausdruck des Textbildschirmes, vier zusätzliche Basic-Befehle sowie eine Möglichkeit zur Erzeugung fast beliebig großer Grafik-Textfenster vor. Außerdem zeigen wir Ihnen anhand eines Beispieles, wie man Zahlen aus dem Hexadezimalsystem in ihr dezimales Äquivalent umwandelt – und umgekehrt.

Für alle neu hinzugekommenen C 16- und C 116-Besitzer haben wir hier nochmals die seit Ausgabe 4/86 veröffentlichten Tips und Tricks zum C 16/C 116 zusammengestellt. Sofern Sie dieser kleinen Sammlung sowie der entsprechenden Rubrik im 64'er noch etwas hinzuzufügen haben, würden wir uns natürlich über Ihren Beitrag freuen.

Veränderung des Zeichensatzes

Der Zeichensatz besteht beim C 16 aus nur 128 Zeichen. Die anderen 128 Zeichen sind die inverse Darstellung der ersten 128 Zeichen. Ein Zeichensatz kann im Abstand von 1024 Byte (also 128 Zeichen mit je 8 Byte) im Speicher stehen (zum Beispiel ab den Adressen 12288, 13312, 14336, 15360).

Nun die Befehle, um ihn zu verändern:

1. »POKE 65298, PEEK (65298) AND NOT 4«
2. »POKE 65299, Anfang Zeichensatz/256«

Nach einer Fehlermeldung sind allerdings keine Zeichen mehr zu erkennen. Sie erscheinen erst dann wieder, wenn erneut ein Befehl eingegeben wird.

Listing 1 zeigt ein Programm zum Kopieren des Zeichensatzes vom ROM ins RAM. Startadresse des Zeichensatzes ist 15360.

Farben auf den Farbtasten verändern

Die Farben der Farbtasten (CTRL- oder Commodore-Taste zusammen mit einer Zahlentaste) sind in den Adressen 275 bis 290 gespeichert. Sie können beliebig geändert werden. Die entsprechenden Steuerzeichen bleiben allerdings gleich!

Beispiel: POKE 275,69.
Statt Schwarz liegt auf der ersten Farbtaste jetzt Grün.

Wiederholfunktion der Tasten

POKE 1344,0:	Wiederholfunktion aus, nur »CRSR«, »SPACE«, »INST/DEL« haben Wiederholfunktion
POKE 1344,64	keine Taste hat Wiederholfunktion
POKE 1344,128	alle Tasten haben Wiederholfunktion (Normalzustand)

Tastaturpuffer

Die Adressen 1319 bis 1328 enthalten die Codes der Tasten, die nicht unmittelbar ausgeführt werden können.

239 enthält den Zähler für den Tastaturpuffer, der angibt, wieviel Codes im Tastaturpuffer abgelegt sind.

Hier ein Beispiel, bei dem ein Programm nachgeladen wird (in einem Programm).

```
POKE 1319,13 : POKE 1320,13 : POKE 239,2
PRINT "DLOAD" : CHR$ (34)
PRINT "RUN"
```

Tastaturabfrage

In der Adresse 198 wird der Code der gerade gedrückten Taste abgelegt. Wenn keine Taste gedrückt wird, ist PEEK (198) gleich 64. Mit PEEK (198) können auch die Funktionstasten abgefragt werden.

Abfragen der »SHIFT«-, »Commodore«- und »CTRL«-Tasten

Mit PEEK (1347) können diese Tasten abgefragt werden.

(Ulrich Käfferbitz/tr/bj)

```
10 FORT=0TO56
20 READA
30 POKE15000+T,A
40 NEXT
50 SYS15000
62900 DATA 169,0,141,248,7,133,208,133
62910 DATA 210,169,208,133,209,169,60,133
62920 DATA 211,141,19,255,160,0,177,208
62930 DATA 145,210,200,208,249,230,209,230
62940 DATA 211,165,211,201,64,208,237,169
62950 DATA 192,141,18,255,169,59,133
62960 DATA 52,133,56
62970 DATA 169,246,133,51,133,55,96
```

Listing 1. Verschiebt den Zeichensatz ins RAM

Nützliche Speicherstellen

- WAIT 1,192 wartet darauf, daß eine Taste am Recorder gedrückt wird.
- Der Funktionstastenspeicher steht in den Adressen $0567 bis $05E6
- Die Einsprungadresse für die USR-Funktion steht in den Adressen 1281/1282 ($0501/$0502)

- Basic-Start	43/44 (Low-/Highbyte)
- Anfang der Variablen	45/46 (Low-/Highbyte)
- Anfang der Felder	47/48 (Low-/Highbyte)
- Ende der Felder	49/50 (Low-/Highbyte)
- Stringspeicher Ende	51/52 (Low-/Highbyte)
- Anfang der Strings	53/54 (Low-/Highbyte)
- Basic-Speicher Ende	55/56 (Low-/Highbyte)

- Der Ton-Chip des C 16 läßt sich in Assembler wie folgt programmieren: Soundregister ist Adresse 65297. Jedem Bit kommt eine bestimmte Steuerfunktion zu:

Bit 0 bis 2	= #0 bis #7	Lautstärke = VOL 0 bis VOL 6
Bit 3	= #8	volle Lautstärke (wenn Bit 3 gesetzt ist, sind Bit 0-2 ohne Bedeutung)
Bit 4	= # 16	Stimme 1 an (=1) / aus (=0)
Bit 5	= # 32	Stimme 2 an (=1) / aus (=0)
Bit 6	= # 64	Stimme 2 Rauschen (=1)
Bit 7	= # 128	Bit gesetzt = kein Ton

Der Ton ist in 10 Bit codiert: mögliche Werte: #0 bis #1023

		Low-Byte	High-Byte (0,1)
Frequenzen	Stimme 1	65294	65298
	Stimme 2	65295	65296

- Register 65286: Durch Löschen der Bits 0/1 läßt sich der Bildschirm um bis zu drei Rasterzeilen nach oben verschieben.

Bit 2 gesetzt	Bildschirm vier Rasterzeilen nach unten
Bit 3 gelöscht	Bildschirm vier Rasterzeilen nach unten
Bit 4 gelöscht	Bildschirm ausgeblendet
Bit 5 gesetzt	Char-Speicher verschoben
Bit 6 gesetzt	kein Cursor sichtbar

– Register 65287	Bit 0/1 setzen	Verschieben des Bildschirms um bis zu drei Rasterzeilen nach rechts
	Bit 4 gesetzt	Mehrfarbmodus ein
	Bit 5 gesetzt	Videochip stoppt (keine Bilderzeugung mehr)
	Bit 6 gesetzt	Programmierbare Bildstörung
	Bit 7 gesetzt	38 Zeichen/Zeile
– POKE 65290, 162		sperrt Tastatur
– POKE 65290, 163		entriegelt Tastatur
– Register 65301		Hintergrundfarbe Bit 0-3 (Low-Nibble) Farbe
– Register 65305		Rahmenfarbe Bit 4-6 (High-Nibble) Intensität

(Frank Plachetta/tr/bj)

```
1000  REM HARDCOPY-ROUTINE
1020  OPEN 4,4,0
1030  FOR X=1 TO 42:X$=X$+"-":NEXT X
1040  PRINT#4,X$
1050  FOR X=0 TO 24:Y$="!"
1060  FOR Y=1 TO 40:Y%=PEEK(3071+X*40+Y):IF Y%<32 THEN
      Y%=Y%+64
1070  Y$=Y$+CHR$(Y%):NEXT Y
1080  Y$=Y$+"!":PRINT#4,Y$:NEXT X
1090  PRINT#4,X$:PRINT#4:PRINT#4:PRINT#4:PRINT#
      4:PRINT#4
1100  PRINT#4:PRINT#4:PRINT#4:PRINT#4:CLOSE4:X$=
      "":Y$="":RETURN
```

Listing 2. »Hardcopy-Routine« für C 16/C 116

Windows im Programm

Im Sonderheft zum C 16 wird auf die Möglichkeit hingewiesen, Bildschirmfenster innerhalb eines laufenden Programms zu erstellen. Die beschriebenen Methoden sind jedoch weiter verbesserungsfähig.

Unter Umgehung des Escape-Modus: Die Koordinaten zur Festlegung eines Windows können direkt in die Systemadressen für die aktuelle Bildschirmgröße »gePOKEt« werden.

```
Adresse
2021 ($07E5): unterer Rand (Zeilenkoordinate/»esc-B«)
2022 ($07E6): oberer Rand  (Zeilenkoordinate/»esc-T«)
2023 ($07E7): linker Rand  (Spaltenkoordinate/»esc-T«)
2024 ($07E8): rechter Rand (Spaltenkoordinate/»esc-B«)
```

Ein Beispiel: Mit POKE 2022,2 wird die erste Bildschirmzeile »eingefroren« (zum Beispiel als Titelzeile), indem der obere Rand des aktuellen Bildschirms herabgesetzt wird.

Der Normalzustand (Window = normale Bildschirmgröße) wird wie üblich durch zweimaliges Drücken der Home-Taste beziehungsweise durch PRINT"Home Home" wiederhergestellt.

(Gerd Watza/tr/bj)

Hardcopy-Routine

Wenn Sie einen Drucker an Ihren Computer angeschlossen haben, können Sie mit Hilfe dieser kleinen Unterroutine (Listing 2) den Bildschirminhalt zu Papier bringen.

Aufgerufen wird das Programm durch »GOSUB 1000«. Um den Bildschirminhalt herum wird ein Rand gedruckt.

(Jürgen Hagen/tr/bj)

Programm-Beschreibung

Zeile:	
1020	Eröffnen eines Drucker-Kanals
1030	Erweitern des Feldes X$ auf insgesamt 42 Bindestriche
1040	Drucken des Feldes X$ als obere Umrandung
1050	Schleife zur Erzeugung von 25 Zeilen mit der Initialisierung der linken Umrandung
1060	Schleife zum PEEKen von 40 Zeichen pro Bildschirmzeile
1070	Ende der Zeichenschleife
1080	Ende der Zeilenschleife mit der rechten Umrandung
1090	Drucken des Feldes X$ als untere Umrandung sowie einige Leerzeilen
1100	Drucken weiterer Leerzeilen, Schließen des Druckers, Löschen der Felder X$ und Y$ sowie ein »RETURN« für den Fall, daß in diese Routine durch ein »GOSUB« verzweigt wurde.

Das seltsame Listing

Wer glücklicher Besitzer des C 16-Sonderheftes (3/86) ist, wird sich beim Abtippen des Farbdemos auf Seite 22 etwas gewundert haben: Das Listing befindet sich nämlich in einem absolut »unabtippbaren« Zustand. Unser Umsetzprogramm für die Steuerzeichen (CLR, Farben etc.) hat die Verarbeitung dieses Listings verweigert. Nachfolgend finden Sie nun die fehlerfreie Version (Listing 3).

(tr/bj)

```
10 REM ********************************
20 REM *                              *
30 REM *   (C 16)  FARBDEMO    (116)  *
40 REM *           --------           *
50 REM *   CHRISTIAN QUIRIN SPITZNER  *
60 REM *   GRUBERSTR. 53, 8011 POING  *
70 REM *   TELEFON:       08121/81100 *
80 REM *                              *
90 REM ********************************
100 COLOR0,4
110 COLOR4,4
120 PRINT"{CLR,DOWN,BLACK,10SPACE}F A R B D E M O"
130 PRINT"{10SPACE}===============(DOWN}"
140 PRINT" H{2SPACE}S W R Z P G B G O B G R B H D H"
150 PRINT" E{2SPACE}C E O Y U R L E R R E O L E U E"
160 PRINT" L{2SPACE}H I T A R U A L A A L S A L N L"
170 PRINT" L{2SPACE}W E . N P E U B N U B A U L K L"
180 PRINT" I{2SPACE}A S . . U N . . G . G . G B E G"
190 PRINT" G{2SPACE}R S . . R . . . E . R . R L L R"
200 PRINT" K{2SPACE}Z . . . . . . . . . U . U A B U"
210 PRINT" E{2SPACE}. . . . . . . . . . E . E U L E"
220 PRINT" I{2SPACE}. . . . . . . . . . N . N . A N"
230 PRINT" T{2SPACE}. . . . . . . . . . . . . . U ."
240 PRINT
250 FOR J=7 TO 0 STEP -1
260 :  PRINT"{BLACK}"J;
270 :  FOR I=1 TO 16
280 :  COLOR 1,I,J
290 :  PRINT"{SPACE,RVSON,SPACE,RVOFF}";
300 :  NEXT I
310 :  PRINT
320 NEXTJ
330 GETKEYA$
©64'er
```

Listing 3. »Farbdemo«. Zeigt alle 121 Farben des C16.

Vier nützliche Basic-Befehle

Das Programm »BASICTOOL« (Listing 4) ist ein sogenannter »DATA-Lader«, der vier kurze Maschinenprogramme erzeugt. Hierdurch werden einige zusätzliche Funktionen auf dem C 16 implementiert:

OLD

Mit diesem Befehl kann man Basic-Programme, die unbeabsichtigt durch »NEW« oder einen Reset gelöscht wurden,

wieder lauffähig machen. Er hilft bisweilen auch, wenn Programme fehlerhaft von Kassette geladen wurden.

Aufgerufen wird er einfach mit: SYS 1618

SWAP

Dieser Befehl wurde für den C 64 bereits in der 64'er, Ausgabe 12/85, vorgestellt. Ich habe ihn für den C 16 umgeschrieben.

Er dient dazu, zwei Strings miteinander zu vertauschen. Dies geschieht durch Vertauschen der Stringdeskriptoren, es entsteht also kein »Stringmüll«. Die gefürchtete Garbage Collection wird verhindert.

Es gilt folgender Syntax für den Aufruf: SYS 1569 (A$,B$)

A$ und B$ können hierbei zwei beliebige Stringvariable sein.

BLOAD und BSAVE

Diese Befehle simulieren die entsprechenden Befehle des C 128. Sie dienen dazu, von einem Basic-Programm aus einen ganzen Speicherblock zu laden oder zu speichern.

Man kann so zum Beispiel eine Grafik speichern oder ein Maschinenprogramm nachladen. Auch Programme, die mit dem S-Befehl des eingebauten Monitors gespeichert wurden, können geladen werden.

Es gilt folgende Syntax:

BLOAD: SYS 1536 "name", g, 1

BSAVE: SYS 1548 "name", g, 1, aa, ea+1

Hierbei ist g die Gerätenummer (1 oder 8 für Kassette oder Disk), aa die Anfangs- und ea die Endadresse des Speicherbereichs.

Beispiel: aa = 8192, ea = 16384 zum Speichern der HiRes-Grafik.

(Michael Schmand/tr/bj)

```
120 DATA 20,6B,A8,A5,0A,A6,2B,A4
130 DATA 2C,4C,D5,FF,20,6B,A8,20
140 DATA DE,9D,84,D8,85,D9,20,DE
150 DATA 9D,A6,14,A8,A9,D8,4C,D8
160 DATA FF,20,8E,94,20,2C,93,20
170 DATA 1A,93,A5,64,85,D0,A5,65
180 DATA 85,D1,20,91,94,20,2C,93
190 DATA 20,1A,93,A0,00,B1,D0,85
200 DATA D2,B1,64,91,D0,A5,D2,91
210 DATA 64,C8,C0,03,D0,EF,20,8B
220 DATA 94,60,A9,01,A8,91,2B,20
230 DATA 18,88,20,4B,88,68,68,4C
240 DATA 7A,8A,00,00,00,00,00,00
260 FOR I=1536 TO 1639
270 READ D$:POKE I,DEC(D$):C=C+DEC(D$)
280 NEXT:IF C<>12205 THEN PRINT" DATA FEHL
    ER!":END
460 PRINT" OBJEKTCODE SPEICHERN (J/N)?"
470 GET KEY C$:IF C$<>"J"THEN END
480 PRINT" CASETTE ODER DISK{4SPACE}(C/D)?
    ":GET KEY C$
490 IF C$="D"THEN U=8:ELSE U=1
500 SYS 1548"MINITOOL.OBJ",U,1,1536,1640
```

Listing 4. Vier nützliche Befehle für den C 16/C 116

Beliebige große Grafikfenster

Dieses Maschinenprogramm steuert einen Rasterzeilen-Interrupt, durch den man im »GRAPHIC 2«- oder »GRAPHIC 4«-Modus ein fast beliebig großes Grafikfenster erzeugen kann. Die hierzu notwendige Routine ist bereits im ROM des C 16 vorhanden, denn auch im normalen Split-Screen-Modus muß ja ein Raster-Interrupt stattfinden. Man kann also diese Routine einfach kopieren und ein wenig ändern. Wie dies am einfachsten geschieht, wird im folgenden beschrieben:

1. Geben Sie zunächst die folgende Basic-Zeile ein:

10 SYS 4293

Dies wird der spätere Einsprung in die Initialisierungs-Routine. Achten Sie darauf, daß zwischen »SYS« und der Adresse höchstens ein Space steht, sonst kommen Sie mit dem Speicherplatz nicht hin.

2. Rufen Sie jetzt den Monitor auf und geben Sie ein:

T CECE CE04 100E

Damit wird die Interrupt-Routine ins RAM kopiert. Hierbei bleibt das Low-Byte der Interrupt-Startadresse gleich ($OE), so daß später nur die High-Bytes der Interrupt-Vektoren geändert werden müssen, um den Interrupt auf unsere eigene Routine zu lenken.

3. Machen Sie nun noch vom Monitor aus folgende Änderungen: A 1015 JSR $1060.

Damit wird der Raster-Interrupt auf die eigene Routine umgelenkt.

4. Zur Initialisierung müssen Sie jetzt noch folgendes kurzes Maschinenprogramm eingeben:

```
A 10C5  SEI
A 10C6  LDA#$10    (Interrupt auf eigene Routine lenken)
A 10C8  STA$0313
A 10CB  STA$0315
A 10CE  CLI
A 10CF  LDA#$11    (Basic-Start auf $1100 heraufsetzen)
A 10D1  STA $2C
A 10D3  LDA#$00
A 10D5  STA $1100
A 10D8  PLA
A 10D9  PLA
A 10DA  JMP $8A7B  (NEW/CLR ausführen)
```

5. Verlassen Sie jetzt den Monitor mit »X« und machen Sie im Direktmodus folgende Eingaben:

POKE 4185,129:POKE 4203,131:POKE4238,131

Damit ist das kleinstmögliche Grafikfenster eingestellt.

6. Rufen Sie wieder den Monitor auf. Speichern Sie das fertige Programm mit:

S "GRAFIKFENSTER",8,1001,10DD

Dadurch wird jetzt die vorher eingegebene Basic-Zeile zusammen mit der kopierten Maschinenroutine als zusammenhängendes Programm gespeichert und später als solches auch wieder geladen.

Kassettenbenutzer müssen statt der »8« natürlich eine »1« einsetzen.

Das so gespeicherte Programm können Sie nun einfach mit »LOAD "GRAFIKFENSTER",8« (beziehungsweise mit »LOAD "GRAFIKFENSTER",1 für Datasette) laden und mit »RUN« starten. Dabei wird automatisch der Basic-Start um 256 Byte nach oben gesetzt. Danach können Sie Basic-Programme eingeben oder laden, als ob nichts geschehen wäre. Sobald Sie aber »GRAPHIC 2« oder »GRAPHIC 4« eingeben, sehen Sie, daß jetzt nicht mehr fünf, sondern neun Textzeilen sichtbar werden. Dies wurde durch die POKE-Befehle erreicht, die einfach dafür sorgen, daß bei einer früheren Rasterzeile als sonst in den Textmodus zurückgeschaltet wird. Weitere POKE-Möglichkeiten entnehmen Sie bitte der Tabelle. Sie können damit jederzeit von Basic aus die Größe des Grafikfensters ändern.

Wichtiger Hinweis:

Sowohl bei der Eingabe als auch beim Laden des Programms muß der Basic-Start unbedingt auf seinem Normalwert sein! Ansonsten liegt die am Anfang eingegebene

Basic-Zeile nicht an der richtigen Stelle im Speicher. Drücken Sie also am besten einmal auf den Reset-Knopf. Dies ist besonders wichtig für Besitzer eines Plus/4 oder eines C 16 mit RAM-Erweiterung, da hier beim erstmaligen Einschalten der Grafik der Basic-Start automatisch um 12 KByte heraufgesetzt wird!

Tabelle:

In nachfolgender Tabelle ist aufgeführt, welchen Effekt man erzielt, wenn man durch POKE-Befehle die Größe des Grafikfensters ändern will. Dazu müssen Sie jeweils in die Speicherstellen 4203 und 42038 den Wert von X, und in Speicherstelle 4185 den Wert von X minus 2 POKEn.

X	Anzahl Textzeilen	Wirkung
131	9	kleinstmögliches Grafikfenster
139	8	
147	7	
155	6	
163	5	normal
171	4	
179	3	
187	2	

Ein besonderer Effekt läßt sich erzielen, wenn man den Wert in 4238 auf X plus 1 setzt. Hier wartet nämlich das Interrupt-Programm, bis der Rasterstrahl den eigentlichen Umschaltpunkt erreicht hat. Dadurch wird ein flimmerfreies Umschalten möglich. Erhöht man nun diesen Wert, so wartet der Interrupt mit dem Einschalten des Textbildschirms noch eine Rasterzeile »lang«, obwohl der Grafikbildschirm bereits ausgeschaltet wurde. Resultat ist eine schwarze Trennlinie zwischen Grafik und Text, die entsteht, während sich der Video-Chip im »Niemandsland« zwischen Text und Grafik befindet. (Michael Schmand/tr/bj)

Warum Hexadezimal?

Hier wollen wir einen leicht verständlichen Lösungsweg zur Umrechnung der beiden Zahlensysteme Dezimal und Hexadezimal erarbeiten.

Wir sind von Kind auf an das dezimale Zahlensystem gewöhnt. In der Computertechnik wird jedoch auch häufig das hexadezimale Zahlensystem verwendet. Warum? Um diese Frage zu beantworten, müssen wir etwas ausholen. Wie Sie vielleicht wissen, kennt der Computer im Prinzip nur das binäre Zahlensystem, also die Kombination von 0 und 1. Aus diesen beiden Ziffern wird jede Zahl gebildet. Beispiel:

binär 00001101 = dezimal 13 = hexadezimal D
binär 1100000000000000 = dezimal 49152 = hexadezimal C000

In Fachzeitschriften finden Sie oft Abkürzungen oder andere Schreibweisen, um klarzustellen, welches Zahlensystem gemeint ist. Beispiel:

binär = bin = %
hexadezimal = hex = $
dezimal = dez = #

Die Zeichen %, $ und # werden vor allem von Programmen benutzt, die irgendetwas mit Assembler (Maschinensprache) oder mit dem Betriebssystem des Computers zu tun haben, zum Beispiel Maschinensprache-Monitore. Wenn Sie mehr darüber wissen wollen, empfehle ich Ihnen das 64'er-Sonderheft 8/85 (Assembler für Anfänger und Fortgeschrittene). Doch zurück zu den Zahlensystemen:

Die ersten Computer konnten nur mit binären Werten »gefüttert« werden, eine sehr anstrengende Sache für die Programmierer. Eine wesentliche Vereinfachung ergab das hexadezimale System. Jeweils 4 Binär-Ziffern konnten ersetzt werden durch eine einzige Hex-Ziffer. Beispiel:

bin 1101 = hex D = dez 13
bin 1000 = hex 8 = dez 8

Das liegt daran, daß das binäre Zahlensystem die Basis 2 besitzt, sich jede Zahl also in 2er-Potenzen darstellen läßt, während das hexadezimale System die Basis 16 hat. Da 2 hoch 4 den Wert 16 ergibt, können 4 Binär-Ziffern durch eine Hex-Ziffer ersetzt werden.

Unser C 16 und auch fast alle anderen Heimcomputer sind 8-Bit-Computer. Der C 16 kann 64 KByte Speicherplatz adressieren, das sind 2 hoch 16 = dezimal 65536 = hex FFFF. Sie sehen, überall taucht die 2 auf, beziehungsweise Potenzen von 2. Und das macht die Verwendung von Hex-Zahlen so einfach, weil man sehr oft mit runden Hex-Zahlen zu tun hat. Das hexadezimale Zahlensystem bietet jedoch nicht nur von seiten der Eingabe Vorteile, sondern erlaubt es auch, sich häufig verwendete Kombinationen zu merken (zum Beispiel $A9 = 169 für den Assembler-Befehl »LDA#«, also lade Akkumulator mit dem Wert der im Programmablauf folgenden Speicherzelle. Genaueres finden Sie, wie schon erwähnt, im Assembler-Sonderheft 8/85).

Was ist das Hexadezimalsystem?

In unserem normalen Zahlensystem repräsentiert jede Stelle einer Zahl eine Zehnerpotenz. Ein Beispiel: Die Zahl 4714 läßt sich auch als Summe von Zehnerpotenzen schreiben

$4714 = 4 * 10^3 + 7 * 10^2 + 1 * 10^1 + 4 * 10^0 = 4 * 1000 + 7 * 100 + 1 * 10 + 4 * 1$

Beim Hexadezimalsystem wird nun jede Stelle einer Zahl nicht mehr durch eine Zehner-, sondern durch eine Sechzehnerpotenz repräsentiert. Auch hier wieder ein Beispiel: Die Hexadezimalzahl 0324 bedeutet nichts anderes als $3 * 16^2 + 2 * 16^1 + 4 * 16^0$ $(=3 * 256 + 2 * 16 + 4 * 1)$.

Dies hat aber noch weitere Konsequenzen:

Im Dezimalsystem wird eine Stelle immer von 0 bis 9 (insgesamt 10 Ziffern) durchgezählt, bevor die nächste Stelle um eins erhöht wird. Also

00 01 02 03 04 05 06 07 08 09
10 11 12 13 14...

Im Hexadezimalsystem jedoch wird eine Stelle um 16 Werte erhöht, bevor zur nächsten Stelle ein Wert hinzugefügt wird. Da aber unsere Ziffern von 0 bis 9 dazu nicht ausreichen, wurden zusätzlich die Buchstaben A bis F herangezogen. Sie vertreten die Zahlenwerte 10 bis 15 (von 0 bis 15 sind es 16 Werte!). Es bedeuten:

A = 10
B = 11
C = 12
D = 13
E = 14
F = 15

00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13 14... 1A 1B 1C...

Die Dezimalzahl 10 ist also gleichwertig mit dem Hexadezimalwert A. Damit wären wir auch schon bei der Umrechnung.

Dezimal - Hexadezimal

Wenn wir eine Dezimalzahl in Hexadezimal umrechnen wollen, so bauen wir den Hex-Wert Stelle für Stelle von links nach rechts auf.

Nehmen wir also an, wir möchten die Dezimalzahl 41717 in Hexadezimal umrechnen. Dazu teilen wir sie erst einmal durch 16^3

$41717 : 16^3 = 10{,}1848145$

Uns interessiert hier nur die Vorkommastelle 10. Sie ist gleichbedeutend mit dem Hex-Wert **A**. Er bildet die letzte Stelle unserer Hexadezimalzahl (die Zählung beginnt rechts und endet links).

Nun müssen wir von unserer Dezimalzahl $10 * 16^3$ abziehen.

Also $41717 - 10 * 16^3 = 757$

Um die nächsten Stellen unserer Hex-Zahl zu erhalten, führen wir diese Prozedur nun noch mit 16^2 und 16^1 durch:

$757 : 16^2 = 2{,}9703125$ (**=2**)
$757 - 2 * 16^2 = 245$
$245 : 16^1 = 15{,}3125$ (**=F**)
$245 - 15 * 16^1 =$ **5**

Als endgültige Umrechnung der Zahl 41717 ins Hexadezimalsystem erhalten wir also **$A2F5**.

Hexadezimal - Dezimal

Diese Umrechnung ist schon wesentlich einfacher. Um die Hex-Zahl $A2F5 wieder zurückzurechnen, geht man wie folgt vor:

$A * 16^3 + 2 * 16^2 + F * 16^1 + 5 * 16^0$

Da man aber mit den Buchstaben A und F nicht rechnen kann, müssen diese als Dezimalzahlen angegeben werden.

$10 * 16^3 + 2 * 16^2 + 15 * 16^1 + 5 * 16^0$

Wenn Sie dies auf Ihrem C 16 einmal ausrechnen, so werden Sie als Ergebnis wieder die Zahl 41717 erhalten!

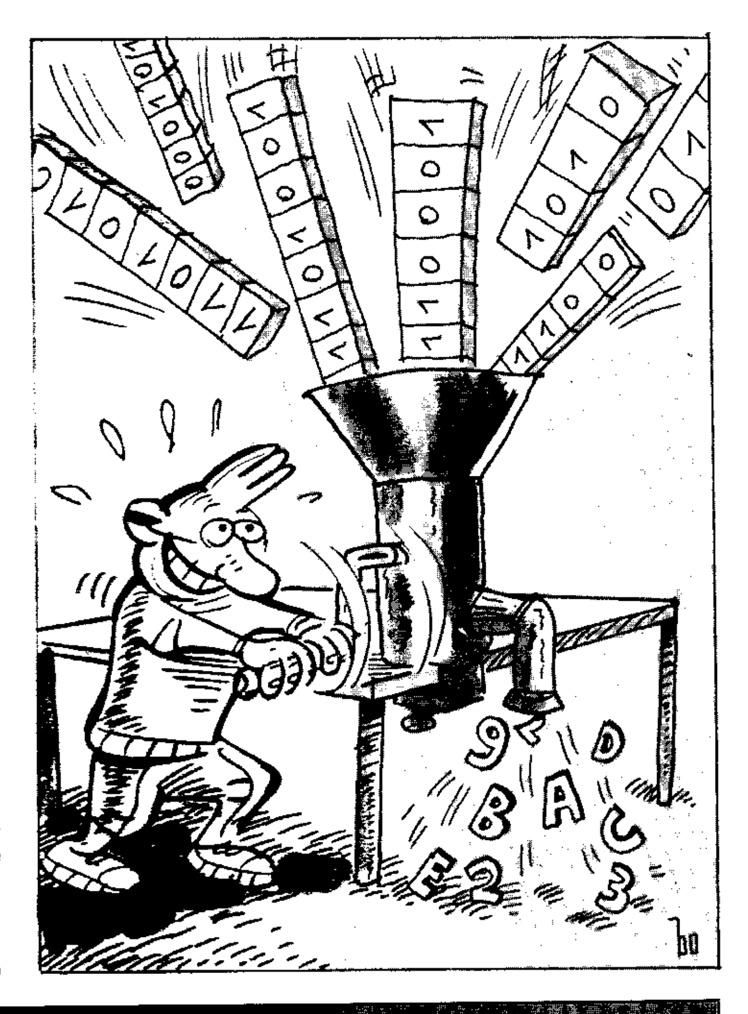
Als Abschluß unseres kleinen Kurses könnten Sie einmal versuchen, ein Basic-Programm zu schreiben, das diese Berechnungen ausführt.

Kleine Merkhilfe

Gerade wenn man anfängt, sich mit dem hexadezimalen Zahlensystem zu beschäftigen (und dafür gibt es einige gute Gründe), ist man froh über ein paar kleine Eselsbrücken. Am Anfang kommt man immer wieder ins Stolpern, weil man sich nicht so schnell an den Dezimalwert von zum Beispiel Hex C erinnern kann. Dies geht wesentlich leichter, wenn man die Anfangsbuchstaben der beiden Systeme gegenüberstellt.

C = **Z**wölf
D = **D**reizehn
F = **F**ünfzehn

(bj)



Depot-Händler

Tragen Sie Ihre Buchbestellung auf eine Postkarte ein und schicken diese an einen Depothändler in Ihrer Nähe oder an Ihren Buchhändler.

Buchhandlung Herder, Kurfürstendamm 69
1000 Berlin 15, Tel. (030) 8835002,
BTX *921782#
Computare Fachbuchhandlung, Keithstraße 18
1000 Berlin 30, Tel. (030) 2139021
Thalia Buchhaus, Große Bleichen 19
2000 Hamburg 36, Tel. (040) 3005050
Boysen + Maasch, Hermannstraße 31
2000 Hamburg 1, Tel. (040) 3005050
Electro-Data, Wilhelm-Heidsiek-Straße 1
2190 Cuxhaven, Tel. (04721) 51288
Buchhandlung Muehlau, Holtenauer Straße 116
2300 Kiel, Tel. (0431) 85085
ECL, Norderstraße 94-96
2390 Flensburg, Tel. (0461) 28181
Buchhandlung Weiland, Königstraße 79
2400 Lübeck, Tel. (0451) 160060
Buchhandlung Storm, Langenstraße 10
2800 Bremen 1, Tel. (0421) 321523
Buchhandlung Lohse-Eissing, Marktstraße 38
2940 Wilhelmshaven, Tel. (04421) 41687
Buchhandlung Schmorl u. v.-Seefeld,
Bahnhofstraße 13
3000 Hannover 1, Tel. (0511) 327651
Buchhandlung Graff, Neue Straße 23
3300 Braunschweig, Tel. (0531) 49271
Deuerlich'sche Buchhandlung, Weender Straße 33
3400 Göttingen, Tel. (0551) 56868
Buchhandlung an der Hochschule,
Holländische Straße 22
3500 Kassel, Tel. (0561) 83807
Stern Verlag, Friedrichstraße 24-26
4000 Düsseldorf, Tel. (0211) 373033
Buchhandlung Baedeker, Kettwiger Straße 33-35
4300 Essen 1, Tel. (0201) 221381
Regensberg'sche Buchhandlung, Alter Steinweg 1
4400 Münster, Tel. (0251) 40541-5
Buchhandlung Acker, Johannisstraße 51
4500 Osnabrück, Tel. (0541) 28488
Buchhandlung Brockmeyer,
Querenburger Höhe 281/Unicenter
4630 Bochum, Tel. (0234) 701360
Buchhandlung Meier + Weber, Warburger Straße 98
4790 Paderborn, Tel. (05251) 63172
Buchhandlung Phönix GmbH, Oberntorwall 25
4800 Bielefeld 1, Tel. (0521) 58306-38
Buchhandlung Gonski, Neumarkt 24
5000 Köln 1, Tel. (0221) 210528
Mayer'sche Buchhandlung, Ursulinerstraße 17-19
5100 Aachen, Tel. (0241) 4777-136
Buchhandlung Behrendt, Am Hof 5a
5300 Bonn 1, Tel. (0228) 658021
Buchhandlung Cusanus, Schloßstraße 12
5400 Koblenz, Tel. (0261) 36239
Akad. Buchhandlung Interbook, Fleischstraße 61-65
5500 Trier, Tel. (0651) 43596
Buchhandlung W. Finke, Kipdorf 32
5600 Wuppertal 1, Tel. (0202) 454220
Buchhandlung Balogh, Sandstraße 1
5900 Siegen, Tel. (0271) 55298-9
Buchhandlung Naacher, Steinweg 3
6000 Frankfurt 1, Tel. (069) 298050
Buchhandlung Wellnitz, Lautenschlägerstraße 4
6100 Darmstadt, Tel. (06151) 76548
Buchhandlung Feller + Gecks, Friedrichstraße 31
6200 Wiesbaden, Tel. (06121) 304911
Ferber'sche UNI-Buchhandlung, Seltersweg 83
6300 Gießen, Tel. (0641) 12001
Sozialwissenschaftliche Fachbuchhandlung,
Friedrichstraße 24
6400 Fulda, Tel. (0661) 75077
Albertis-Hofbuchhandlung, Langstraße 47,
6450 Hanau, Tel. (06181) 24301
Gutenberg Buchhandlung, Große Bleiche 29
6500 Mainz, Tel. (06131) 37011
Buchhandlung Bock + Seip, Futterstraße 2
6600 Saarbrücken, Tel. (0681) 30677
Buchhandlung Wilhelm Hofmann,
Bismarckstraße 98
6700 Ludwigshafen, Tel. (0621) 516001
Buchhandlung Loeffler, B 1,5
6800 Mannheim 1, Tel. (0621) 28912
Buchhandlung Stehn, Bahnhofstraße 13
7000 Stuttgart 50, Tel. (0711) 561476
Osiandersche Buchhandlung, Sindelfinger Allee 25
7030 Böblingen
Buchhandlung am Markt, Kramstraße 6
7100 Heilbronn, Tel. (07131) 68682
UNI Buchhandlung Kellner + Moessner,
Kaiserstraße 18
7500 Karlsruhe, Tel. (0721) 691436
Osiandersche Buchhandlung, Wilhelmstr. 12
7400 Tübingen, Tel. (07071) 51761
Osiandersche Buchhandlung, Kaiserpassage 8
7410 Reutlingen

Buchhandlung Roth, Hauptstraße 45
7600 Offenburg, Tel. (0781) 22097
Rombach Center, Bertholdstraße 10
7800 Freiburg, Tel. (0761) 49091
Fachbuchhandlung Hofmann, Hirschstraße 4
7900 Ulm, Tel. (0731) 60949
Schauties Elektronik, Bachstraße 52
7980 Ravensburg, Tel. (0751) 26138
Buchhandlung Hugendubel, Marienplatz
8000 München 2, Tel. (089) 2389-1
Computerbücher am Obelisk, Barerstraße 32-34
8000 München 2, Tel. (089) 282383
Pele's Computerbücher, Schillerstraße 17
8000 München 2, Tel. (089) 555229
Universitätsbuchhandlung Lachner,
Theresienstraße 43
8000 München 2, Tel. (089) 521340
Buchhandlung Schönhuber, Theresienstraße 6
8070 Ingolstadt, Tel. (0841) 33146/47
Computerstudio Gertrud Friedrich, Ludwigstraße 3
8220 Traunstein, Tel. (0861) 14767
Buchhandlung Pustet, Kl. Exerzierplatz 4
8390 Passau, Tel. (0851) 56945
Buchhandlung Pustet, Gesandtenstraße 6
8400 Regensburg, Tel. (0941)53061
Buchhandlung Dr. Büttner, Adlerstraße 10-12
8500 Nürnberg, Tel. (0911) 232318
Computer-Center-Burger, Leimitzer Straße 11-13
8670 Hof, Tel. (09281) 40075
Sortiments- u. Bahnhofsbuchh. J. Strykowski,
Bahnhofplatz 4
8700 Würzburg, Tel. (0931) 54389
Buchhandlung Pustet, Grottenau 4
8900 Augsburg, Tel. (0821) 35437
Kemptener Fachsortiment, Salzstraße 30
8960 Kempten, Tel. (0831) 14413

Schweiz:
Buchhandlung Francke AG, Neuengasse 43,
Von-Werdt-Passage
3001 Bern, Tel. (031) 221717
Buchhandlung Scherz, Marktgasse 25
3011 Bern, Tel. (031) 226837
Buchhandlung Meissner, Bahnhofstrasse 41
5000 Aarau, Tel. (064) 247151
Bücher Balmer, Neugasse 12
6300 Zug, Tel. (042) 214141
Buchhandlung Enge, Bleicherweg 56
8002 Zürich, Tel. (01) 2012078
Buchhandlung Orell Füssli, Pelikanstrasse 10
8022 Zürich, Tel. (01) 2118011
Freihofer AG, Wissenschaftliche Buchhandlung,
Universitätsstrasse 11
8033 Zürich, Tel. (01) 3634282
Buchhandlung am Rösslitor, Webergasse 5
9001 St. Gallen, Tel. (071) 228726

Österreich:
Morawa & Co. Wollzeile 11
1010 Wien, Tel. (0222) 947641
Computer Buch Shop Karl Fegerl, Heinertstraße 3
1020 Wien, Tel. (0222) 245368
Johann Reisinger, Hauptplatz 30, Kirchenstraße 3
3302 Amstetten, Tel. (07472) 2576-0
Helmut Lainer, Obere Landstraße 8
3500 Krems, Tel. (02732) 2818
R. Pirngruber, Landstraße 34
4020 Linz, Tel. (0732) 272834
Buchhandlung Schachtner, Stadtplatz 28
4840 Vöcklabruck, Tel. (07672) 3467
R. Regelsberg, St.-Julien-Straße 2
5020 Salzburg, Tel. (0662) 73573
Tyrolia, Maria-Theresien-Straße 15
6010 Innsbruck, Tel. (05222) 24944
Wagner'sche Universitätsbuchhandlung,
Museumstraße 4
6010 Innsbruck, Tel. (05222) 22316
Buchhandlung Leykam, Stemplergasse 3
8010 Graz, Tel. (0316) 76676-0
Jos. A. Kienreich, Sacherstraße 6
8010 Graz, Tel. (0316) 76441
Volksbuchhandlung, Radetzkystraße 7
8010 Graz, Tel. (0316) 79388



Unternehmensbereich Buchverlag
Hans-Pinsel-Straße 2, 8013 Haar bei München

Impressum

Herausgeber: Carl-Franz von Quadt, Otmar Weber

Chefredakteur: Michael Scharfenberger
Stellv. Chefredakteur: Albert Absmeier
Koordination: Georg Klinge
Redaktion: Achim Hübner (ah), Karsten Schramm (ks), Herbert Buckel (bj), Harald Meyer (hm), Dieter Mayer (dm), Markus Ohnesorg (og), Norbert Jungmann (nj), Gerd Donaubauer (do), Thomas Röder (tr), Boris Schneider (bs), Gottfried Knechtel (kn)

Titelfoto: Jens Jancke

Titelgestaltung: Heinz Rauner Grafik-Design

Layout:
Leo Eder (Ltg.), Sigrid Kowalewski (Cheflayouterin), Rolf Raß, Katja Milles

Produktionsleiter: Klaus Buck

Auslandsrepräsentation:
Schweiz: Markt & Technik Vertriebs AG, Kollerstr. 3, CH-6300 Zug, Tel. 042-415656, Telex: 862329
USA: M&T Publishing Inc.; 501 Galveston Drive Redwood City, CA 94063 Telefon: (415) 366-3600

Manuskripteinsendungen: Manuskripte und Programmlistings werden gerne von der Redaktion angenommen. Sie müssen frei sein von Rechten Dritter. Sollten sie auch an anderer Stelle zur Veröffentlichung oder gewerblichen Nutzung angeboten werden, so muß dies angegeben werden. Mit der Einsendung von Manuskripten und Listings gibt der Verfasser die Zustimmung zum Abdruck in von der Markt & Technik Verlag AG herausgegebenen Publikationen und zur Vervielfältigung der Programmlistings auf Datenträger. Mit der Einsendung von Bauanleitungen gibt der Einsender die Zustimmung zum Abdruck in von Markt & Technik Verlag AG verlegten Publikationen und dazu, daß Markt & Technik Verlag AG Geräte und Bauteile nach der Bauanleitung herstellen läßt und vertreibt oder durch Dritte vertreiben läßt. Honorare nach Vereinbarung. Für unverlangt eingesandte Manuskripte und Listings wird keine Haftung übernommen.

Marketingleiter: Hans Hörl (114)

Vertriebsleiter: Helmut Grünfeldt (189)

Anzeigenverwaltung und Disposition: Michaela Hörl

Verlagsleiter M&T-Buchverlag: Günther Frank (212)

Druck: SOV St. Otto-Verlag GmbH, Laubanger 23, 8600 Bamberg

Preis: Das Einzelheft kostet DM 14,–

Vertrieb Handelsauflage: Inland (Groß-, Einzel- und Bahnhofsbuchhandel) sowie Österreich und Schweiz: Pegasus Buch- und Zeitschriften-Vertriebs GmbH, Hauptstätter Straße 96, 7000 Stuttgart 1, Telefon (0711) 64830

Verantwortlich:
Für redaktionellen Teil: Michael Scharfenberger
Für Anzeigen: Britta Fiebig

Redaktions-Direktor: Michael M. Pauly

Vorstand: Carl-Franz von Quadt, Otmar Weber

Anschrift für Verlag, Redaktion, Vertrieb, Anzeigenverwaltung und alle Verantwortlichen:
Markt & Technik Verlag Aktiengesellschaft, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 4613-0, Telex 5-22052

Aktionäre die mehr als 25% des Kapitals halten: Otmar Weber, Ingenieur, München; Carl-Franz von Quadt, Betriebswirt, München; Aufsichtsrat: Dr. Robert Dissmann (Vorsitzender), Karl-Heinz Fanselow, Eduard Heilmayr

DIE BESTEN SPIELE FÜR IHREN C16:

BONGO CONSTRUCTION SET

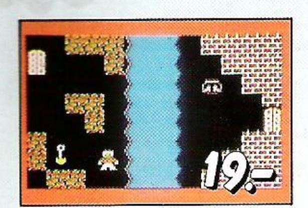
Begleiten Sie Bongo, die Supermaus, in sechs verschiedenen Bildern auf ihrer Suche nach den geraubten Diamanten der Prinzessin. Als einmalige Neuheit können Sie Ihre Bilder selber zusammenbauen, spielen und abspeichern; dadurch bieten sich nahezu unbegrenzte Möglichkeiten. Für 1 oder 2 Spieler; nur mit Joystick.



GALAXY

Schnelles und abwechslungsreiches Weltraumspiel mit achtzehn verschiedenen Phasen und vielen interessanten Gegnern. Gute Reaktionen und genaue Schüsse sind dabei erforderlich, denn Fehlschüsse kosten Sie nur unnötig Energie. Für 1 oder 2 Spieler; Steuerung wahlweise mit Joystick oder Tastatur.





GHOST TOWN

Suchen Sie die verborgene Schatztruhe in einer unheimlichen Geisterstadt. Dabei wandern Sie durch neunzehn Bilder in wunderschöner hochauflösender Farbgrafik. Ein Grafik-Adventure für alle Tüftler und Knobler. Steuerung mit Joystick oder Tastatur.



GRANDMASTER

Das legendäre Schachprogramm in einer vollwertigen Version für den C-16 mit überragender Spielstärke und viel Komfort (u.a. Zugzurücknahme, Zugvorschlag, Wahl der Bildschirmfarben, Schachuhren, Autoplay-Demomodus, usw.). Bedienung mit Tastatur.

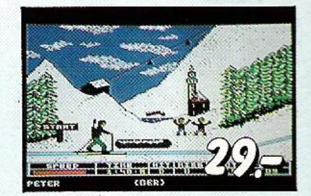
TOM

Durchsuchen Sie mit Ihrem Abenteurer Tom die Pyramide von Manilo nach einem Schatz. Ein Arcade-Adventure mit sage und schreibe 178 (einhundertachtundsiebzig!) Hires-Bildern, die ohne Nachladen im Speicher sind. Steuerung mit Joystick oder Tastatur.



WINTER OLYMPIADE

Endlich ist es soweit: spielen Sie im Familien- oder Freundeskreis sechs verschiedene Disziplinen mit bestechender Grafik: Biathlon, Slalom, Eisschnelllauf, Bobfahren, Skispringen und Abfahrt. Natürlich komplett mit Eröffnungszeremonie, Wahl der Landesfarben, usw. Für 1 bis 4 Mitspieler; Joystick(s) erforderlich.



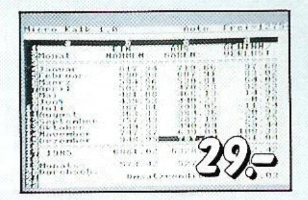
IHR C16 KANN MEHR ALS NUR SPIELEN:





MICRO DATEI

Ein universelles Dateiprogramm für beliebige Daten (z.B. Adressen, Schallplatten, Videos, usw.). Sie können Ihre Datensätze eingeben, ändern, sortieren, ausdrucken, abspeichern, usw. Eine Speichererweiterung ist empfehlenswert.

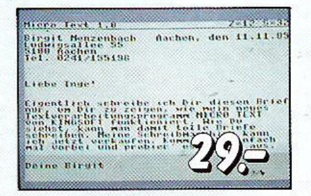


MICRO KALK

Jetzt können Sie alle Kalkulationen durchführen, die tagtäglich anfallen, z.B. Führung einer Haushaltskasse, Einkauf-/Verkauf-Erlös, usw. Schauen Sie sich an, warum Tabellenkalkulationen eine Hauptanwendung der "großen" Personal Computer ist. Eine Speichererweiterung ist empfehlenswert.

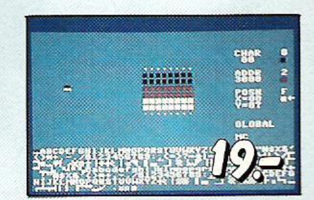
MICRO TEXT

Leistungsfähiges und einfach zu bedienendes Textverarbeitungsprogramm mit vielen Profi-Funktionen, z.B. Flattersatz (linksbündig), Blocksatz (rechtsbündig), Suchen, halbautomatischer Worttrennung, Textbausteine, usw. Der Textspeicher umfaßt in der Grundversion ca. 6000 Zeichen (mehr als eine DIN A4-Seite).



GRAFIK DESIGNER

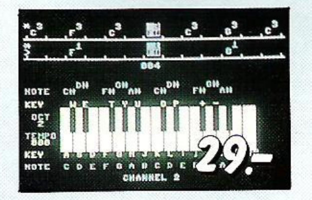
Entwerfen Sie mit diesem Super-Editor Ihren eigenen Zeichensatz. Neben den üblichen Funktionen bietet dieses Programm so tolle Sachen wie: Drehen, Spiegeln, Invertieren, Multicolour-Modus, Scrollen, Füllen, usw. Selbst Animation von mehreren Zeichen ist möglich! Alle unsere Spiele wurden hiermit geschrieben!





PAINT BOX

Ein Mal- und Zeichenprogramm der Spitzenklasse mit vielfältigen Möglichkeiten, z.B. Linien, Strahlen, Rahmen, Rechteck, Kreis, Scheibe, Laden und Abspeichern, Füllen von Flächen, acht verschiedene Pinsel, usw., alles natürlich in 121 Farben. Bedienung mit Tastatur oder Joystick.



MUSIC MASTER

Ein Musikprogramm mit tollen Möglichkeiten, die Ihren C-16 in einen fantastischen Synthesizer und Sequenzer verwandeln. Die mitgelieferte Demo zeigt, daß Ihr C-16 ein fast ebenso guter Musiker ist wie ein C-64. Sie können Ihre Musikstücke aber nicht nur spielen und abspeichern, sondern auch einfach in Ihre eigenen Programme einbauen!

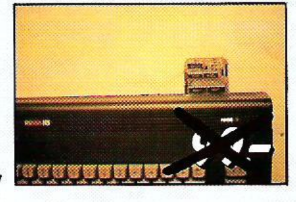
TURBO TAPE

Machen Sie Ihrer Datassette Beine: mit TURBO TAPE können Sie Programme bis zu 28K Länge genauso schnell speichern und laden wie mit einer Floppy! Ein Programm von 12K Länge wird jetzt in 48 Sekunden statt in 6:42 Minuten geladen, also mehr als 8mal so schnell. Das abgespeicherte Programm kann später von jedem C-16 wieder mit Turbo-Geschwindigkeit geladen werden; außerdem bleibt der Bildschirm während des Ladens eingeschaltet! 19.-

ERWEITERN SIE DIE MÖGLICHKEITEN IHRES C16:

16K-RAM

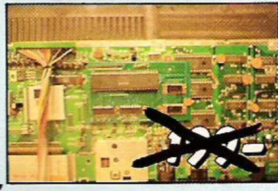
Mit dieser Erweiterung können Sie Ihren C-16 oder C-116 auf 32K-RAM ausbauen, das heißt "28661 BYTES FREE" für Basic-Programme. Endlich haben Sie auch noch genug Speicherplatz zur Verfügung, wenn Sie im hochauflösenden Grafik-Modus arbeiten. Einfach hinten einstecken – einschalten – fertig!



jetzt 79.-

64K-RAM

Durch den Einbau dieser Platine können Sie Ihren C-16 auf die maximale Speicherkapazität von 64K aufrüsten, d.h. "60671 BYTES FREE" in Basic. Der Einbau erfolgt in wenigen Minuten einfach durch Austausch eines Bauteils. Es sind KEINE LÖTARBEITEN erforderlich! Um eine häufig gestellte Frage vorab zu beantworten: auch mit diesem 64K ist der C-16 nicht kompatibel zum C-64!



jetzt 139.-

Alle aufgeführten Artikel werden selbstverständlich mit ausführlicher deutscher Anleitung geliefert. Die Programme laufen auch auf den Computern C-116 und PLUS/4; die 64K-RAM ist aus Platzgründen aber nur im C-16 zu verwenden! Alle Preise verstehen sich als unverbindliche Preisempfehlung zzgl. 5,- DM Porto & Verpackung; der Versand erfolgt ausschließlich per Nachnahme.

SPITZEN - SOFTWARE MADE IN GERMANY

KINGSOFT

F. Schäfer · Schnackebusch 4 · 5106 Roetgen · ☎ 02408/51 19

DAS GOLDENE KINGSOFT ANGEBOT

für Ihren Commodore C-16:

DAS GROSSE C-16 BUCH

Hier erfahren Sie alles, was Sie brauchen, um die Möglichkeiten Ihres C-16 voll auszunutzen. Sämtliche wichtigen Bereiche, wie z.B. Grafik, Sound, Maschinensprache werden ausführlich behandelt und mit vielen Beispielen erläutert. Dieses Buch ist für jeden Commodore-16 Besitzer unentbehrlich.

Einzelpreis: 29.-



Das große C-16 Buch

Schäfer

COMMODORE 16 116 PLUS/4

JOYSTICK

Original Commodore-Joystick im eleganten und handlichen Design für viele Stunden Spielspaß; steigert die Freude bei praktisch allen Spielen enorm. Direkt anschlußfertig für Ihren C-16.

Einzelpreis: 29.-



PLUS-PAKET

Die ideale Spielesammlung für alle Einsteiger, bestehend aus 4 absoluten Top-Programmen

GRANDMASTER, spielstarkes Schachprogramm mit viel Komfort
TOM, faszinierendes Arcade-Adventure mit 178 (!) Bildern
GALAXY, schnelles und abwechslungsreiches Weltraumspiel
GHOST TOWN, spannendes Grafik-Adventure für die ganze Familie

Einzelpreis: 39.-

Sie sparen 28.-

Alle 3 Teile zusammen statt ~~97.-~~ nur



KINGSOFT
F. Schäfer · Schnackebusch 4
5106 Roetgen · Tel. 02408/5119